NASA Technical Memorandum 103766

GRID3D-v2: An Updated Version of the GRID2D/3D Computer Program for Generating Grid Systems in Complex-Shaped Three-Dimensional Spatial Domains

E. Steinthorsson and T.I-P. Shih
Carnegie Mellon University
Pittsburgh, Pennsylvania

and

R.J. Roelke
Lewis Research Center
Cleveland, Ohio

June 1991

NASA

CONTENTS

GRID3D-v2: An Updated Version of the GRID3D Computer Program for Generating Grid Systems in Complex-Shaped Three-Dimensional Spatial Domains

E. Steinthorsson and T. I-P. Shih
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

R. J. Roelke
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

1.0 INTRODUCTION

GRID2D/3D (refs. 1 and 2) is a powerful grid-generation package, capable of generating grid systems for complicated geometries in both two and three dimensions. This package, which employs algebraic grid-generation techniques, is computationally efficient and easy to use. Nonetheless, when the geometry is unusually complex (e.g., see fig. 1-1), the partitioning of the geometry into zones or blocks that are suitable for GRID2D/3D becomes very cumbersome. In order to make GRID2D/3D more versatile and readily applicable to geometries like the one shown in figure 1-1, a number of modifications were made to the package. These modifications are as follows:

(1) specification of boundary curves has been made more flexible;
(2) control over grid-point distribution has been increased;
(3) new interpolating functions based on tension splines have been added; and
(4) control over orthogonality at boundary surfaces has been increased.

GRID2D/3D is made up of two programs, GRID2D and GRID3D. GRID2D generates grid systems for two-dimensional (2-D) spatial domains, and GRID3D generates grid systems for three-dimensional (3-D) domains. The aforementioned modifications were made to GRID3D only. In this report, the original version of GRID3D as reported in references 1 and 2 will be referred to as GRID3D-v1. The new modified version will be referred to as GRID3D-v2.

In the remainder of this report, the theory and method behind the modifications are described, and the use of GRID3D-v2 is explained and illustrated by an example.

2.0 THEORY AND METHODOLOGY

In this section, the theory and methodology behind the modifications that were incorporated into GRID3D-v2 are described. First, specification of boundary curves is discussed, and a new approach for controlling distribution of grid points is explained. Next, the functional forms of new tension-spline based connecting curves are derived, and the properties of these functions examined. Finally, control of orthogonality of grid lines at boundaries is discussed.

Note that throughout this section, the reader is assumed to be familiar with the theory behind GRID3D-v1 which is described in reference 1.

2.1 Boundary Curves and Distribution of Grid Points

When generating 3-D grid systems with GRID3D-v1 or GRID3D-v2, we assume that the geometry of the spatial domain for which a grid is to be generated is completely described by the edge curves of the boundary surfaces (as used here, the edge curves are the four plane or twisted curves that define the boundary of a surface). The algorithms used in GRID3D-v1 and GRID3D-v2 were designed to construct grid systems from the edge curves; however, these algorithms differ in the way they specify edge curves and control grid-point distributions. This difference is described in this section.

In GRID3D-v1, the following approach is used:

(1) Each edge curve is given by specifying a set of node points that lie on the curve. From these node points, a parametric description of the curve is constructed by using tension-spline interpolation.

(2) The distribution of grid points within the domain (including edge curves) is controlled by specifying three one-dimensional stretching functions -- one for each family of grid lines (i.e., one for ξ-grid lines, one for η-grid lines, and one for ζ-grid lines).

Although this approach provides flexibility in generating grid systems within complex-shaped spatial domains, it is inadequate in some cases. For example, it does not allow for edge curves to have derivative discontinuities such as those at cusps. Also, it does not provide adequate control over distribution of grid points in regions where geometry changes appreciably. As a specific example, the distribution of grid points on all constant-ξ surfaces must be the same and cannot, even with partitioning, be made to vary from one section of the grid system to the next.

To overcome the shortcomings of GRID3D-v1, the following approach was used in GRID3D-v2:

(1) Each edge curve is defined by specifying the location of the grid points on the curve. This can be done by either one of the following two procedures: (a) Specify a set of node points that lie on the edge curve for interpolation with a tension spline, and specify a stretching function that controls where on the edge curve the grid points lie. (b) Specify the grid points directly (in this case, a stretching function is not specified by the user, rather it is calculated based on arc length as will be shown herein).

(2) The grid point distribution along grid lines of a given family is that obtained by the bilinear interpolation of the stretching functions used for the edge curves belonging to that family.

The specification of edge curves as described under (1(a)) is self explanatory, but the calculation of stretching functions as described under (1(b)) and (2) requires further explanation. For illustration, consider the ζ–family of grid lines. Suppose all edge curves belonging to this family of grid lines are specified by using procedure (1(a)), and suppose the stretching functions $\hat{\zeta}_{00}(\zeta)$, $\hat{\zeta}_{10}(\zeta)$, $\hat{\zeta}_{01}(\zeta)$ and $\hat{\zeta}_{11}(\zeta)$ describe the distribution of the grid points on the edge curves located at ($\xi = 0, \eta = 0$), ($\xi = 1, \eta = 0$), ($\xi = 0, \eta = 1$) and ($\xi = 1, \eta = 1$), respectively. In GRID3D-v2, the stretching function for a ζ-grid line at any ξ–η location is given by the bilinear interpolation of the stretching functions at the edge curves; namely

$$\hat{\zeta}_{\xi\eta}(\zeta) = \left[\hat{\zeta}_{00}(\zeta)\,(1-\xi) + \hat{\zeta}_{10}(\zeta)\,\xi\right](1-\eta) + \left[\hat{\zeta}_{01}(\zeta)\,(1-\xi) + \hat{\zeta}_{11}(\zeta)\,\xi\right]\eta \tag{2.1}$$

Now, instead of all edge curves being defined by (1(a)), suppose that the edge curve at $\xi = 0$ and $\eta = 0$ is defined by using procedure (1(b)); that is, by specifying the grid point coordinates directly. In order to use equation (2.1) for this case, a stretching function must be calculated for the edge curve. Since the stretching function is needed only at the grid point-locations along the edge curve, it can be calculated by using approximate arc length as follows:

$$\hat{\zeta}_{00}(\zeta_k) = 0 \qquad\qquad k=1 \tag{2.2a}$$

and

$$\hat{\zeta}_{00}(\zeta_k) = \frac{d_k}{d_{KL}} \qquad\qquad k=2,3,4,...,KL \tag{2.2b}$$

where

$$d_k = \sum_{n=2}^{k} \left[(x_n-x_{n-1})^2 + (y_n-y_{n-1})^2 + (z_n-z_{n-1})^2\right]^{1/2} \tag{2.2c}$$

and

$$\zeta_k = (k-1)\,\Delta\zeta \qquad\qquad \Delta\zeta = 1/(KL-1) \tag{2.2d}$$

In equation (2.2), k = 1,2,3,...,KL denotes the grid points on the edge curve; KL is the total number of grid points on the curve; and x_n, y_n and z_n are the x-, y- and z-coordinates of the n-th grid point on the curve.

The same approach as that just described for the ζ-family of grid lines is employed to determine the stretching functions for the ξ- and η-families of grid lines. This approach gives a smooth distribution of grid points throughout the domain.

Finally, note that all stretching functions available in GRID3D-v1 for controlling the distribution of grid points in the entire domain are available in GRID3D-v2 for controlling the distribution of grid points along edge curves defined by using procedure (1(a)) (i.e., by specifying a set of node points that lie on the edge curve and interpolating with a tension spline). In GRID3D-v2, a stretching function that allows asymmetric clustering of grid points along the edge curve was added. The new stretching function, which was developed by Vinokur (ref. 3), is given here.

Let $t \in [0,1]$ be normalized distance or any monotonic parameter along a curve, and let $\xi \in [0,1]$ be the computational coordinate along which grid points are equally spaced. Two user controlled parameters, s_0 and s_1, and two secondary parameters, A and B, are defined as

$$s_0 = \frac{d\xi(t=0)}{dt} \qquad \text{and} \qquad s_1 = \frac{d\xi(t=1)}{dt} \qquad s_0,\, s_1 > 0 \tag{2.3}$$

$$A = \sqrt{s_0/s_1} \qquad \text{and} \qquad B = \sqrt{s_0\, s_1} \tag{2.4}$$

In terms of these parameters, the functional form of the stretching function can be written as

$$t(\xi) = \frac{u(\xi)}{A + (1-A)\,u(\xi)} \tag{2.5}$$

where the function $u(\xi)$ depends on the value of the parameter B, as shown in the following equations.

If B > 1.001, then

$$u(\xi) = \frac{1}{2} + \frac{\tanh[\Delta y\,(\xi - \frac{1}{2})]}{2\tanh[\Delta y/2]} \tag{2.6a}$$

where Δy is obtained from the relation

$$B = \frac{\sinh(\Delta y)}{\Delta y} \tag{2.6b}$$

If $B < 0.999$, then

$$u(\xi) = \frac{1}{2} + \frac{\tan[\Delta x\,(\xi - \frac{1}{2})]}{2\tan[\Delta x/2]} \tag{2.7a}$$

where Δx is obtained from the relation

$$B = \frac{\sin(\Delta x)}{\Delta x} \tag{2.7b}$$

Finally, if $0.999 \le B \le 1.001$, then

$$u(\xi) = \xi\left[1 + 2\,(B - 1)\,(\xi - \tfrac{1}{2})\,(1 - \xi)\right] \tag{2.8}$$

The amount of clustering produced by the stretching function is controlled by the parameters s_0 and s_1 which are defined by equation (2.3). If s_0 and s_1 are greater than one, then grid points are clustered near the boundaries where $t = 0$ and $t = 1$. The greater s_0 and s_1, the greater is the clustering of grid points near the $t = 0$ and $t = 1$ boundaries, respectively. If s_0 and s_1 are less than one, then the grid spacing is larger near the boundaries than in the interior; the smaller s_0 and s_1, the greater the grid spacing near the boundaries.

Frequently, when using the stretching function given by equations (2.3) to (2.8), we must either solve equation (2.6(b)) for Δy, or solve equation (2.7(b)) for Δx. Vinokur (ref. 3) developed approximate analytical relations for both of these inversion problems as follows:

For equation (2.6(b)), which is used when $B > 1.001$, the approximate inverse when $1.001 < B < 2.7829681$ is

$$\begin{aligned}\Delta y = \sqrt{6\beta}\,(1 &- 0.15\beta + 0.057321429\beta^2 - 0.024907295\beta^3 \\ &+ 0.0077424461\beta^4 - 0.0010794123\beta^5)\end{aligned} \tag{2.9a}$$

where

$$\beta = B - 1 \tag{2.9b}$$

When $B > 2.7829681$,

$$\Delta y = v + (1 + 1/v) \ln(2v) - 0.02041793 + 0.24902722w$$
$$+ 1.9496443w^2 - 2.6294547w^3 + 8.56795911w^4 \qquad \textbf{(2.10a)}$$

where

$$v = \ln (B) \qquad \textbf{(2.10b)}$$

and

$$w = (1/B) - 0.028527431 \qquad \textbf{(2.10c)}$$

For equation (2.7(b)), which is used when $B < 0.999$, the approximate inverse when $0 < B < 0.26938972$ is

$$\Delta x = \pi \, [1 - B + B^2 - (1 + \pi^2 / 6) \, B^3 + 6.794732B^4$$
$$- 13.205501B^5 + 11.726095B^6)] \qquad \textbf{(2.11)}$$

When $0.26938972 < B < 0.999$,

$$\Delta x = \sqrt{6\beta} \, (1 + 0.15\beta + 0.057321429\beta^2 + 0.048774238\beta^3$$
$$- 0.053337753\beta^4 + 0.075845134\beta^5) \qquad \textbf{(2.12a)}$$

where

$$\beta = 1 - B \qquad \textbf{(2.12b)}$$

2.2 Connecting Curves Based on Tension Spline Interpolation

Experience has shown that Hermite interpolation (cubic polynomials) as used in GRID3D-v1 sometimes results in connecting curves with too much curvature. In GRID3D-v2, the Hermite interpolation is replaced by tension-spline interpolation. The most attractive feature of tension-spline interpolation is that as the tension parameter is increased from zero to infinity, the interpolation function varies from being a cubic polynomial to being a linear polynomial. Thus, tension-spline interpolation offers increased control over the shape of the grid lines in the grid system. In this section, a derivation of the tension-spline interpolation function is given for the two- and four-

boundary methods. First, the interpolation for an arbitrary variable is derived. Then, the application to algebraic grid generation is illustrated.

The tension-spline interpolation function is derived as follows: suppose the variable X is a function of the parameter s on an interval [0,1], but only X(0), X(1), X'(0) and X'(1) (X' denotes dX/ds) are known. A tension-spline interpolation of X(s) on the interval [0,1] is sought. A tension-spline interpolation of X(s) is traditionally written in terms of X(0), X(1), X"(0) and X"(1), where $X'' = d^2X/ds^2$, as follows (see, e.g., ref. 4):

$$X(s) = \frac{X''(0)\sinh[\sigma(1-s)]}{\sigma^2 \sinh[\sigma]} + \left(X(0) - \frac{X''(0)}{\sigma^2}\right)(1-s) + \frac{X''(1)\sinh[\sigma s]}{\sigma^2 \sinh[\sigma]} + \left(X(1) - \frac{X''(1)}{\sigma^2}\right)s \quad (2.13)$$

where σ is the tension parameter. By differentiating equation (2.13) and evaluating the resulting equation at the end points s = 0 and s = 1, we obtain

$$X'(0) = -\frac{X''(0)\cosh[\sigma]}{\sigma \sinh[\sigma]} - \left(X(0) - \frac{X''(0)}{\sigma^2}\right) + \frac{X''(1)}{\sigma \sinh[\sigma]} + \left(X(1) - \frac{X''(1)}{\sigma^2}\right) \quad (2.14a)$$

and

$$X'(1) = -\frac{X''(0)}{\sigma \sinh[\sigma]} - \left(X(0) - \frac{X''(0)}{\sigma^2}\right) + \frac{X''(1)\cosh[\sigma]}{\sigma \sinh[\sigma]} + \left(X(1) - \frac{X''(1)}{\sigma^2}\right) \quad (2.14b)$$

The above two simultaneous equations can be solved to give expressions for X"(0) and X"(1) in terms of X(0), X(1), X'(0) and X'(1). Substituting the resulting expressions into equation (2.13) gives

$$X(s) = X(0)h_1(s) + X(1)h_2(s) + X'(0)h_3(s) + X'(1)h_4(s) \quad (2.15)$$

$$h_1(s) = c_1(1-s) + c_2 s + c_2\left(\frac{\sinh[\sigma(1-s)] - \sinh[\sigma s]}{\sinh[\sigma]}\right) \quad (2.16a)$$

$$h_2(s) = c_1 s + c_2\,(1-s) - c_2\left(\frac{\sinh[\sigma(1-s)] - \sinh[\sigma s]}{\sinh[\sigma]}\right) \quad (2.16b)$$

$$h_3(s) = c_3\left((1\text{-}s) - \frac{\sinh[\sigma(1\text{-}s)]}{\sinh[\sigma]}\right) + c_4\left(s - \frac{\sinh[\sigma s]}{\sinh[\sigma]}\right) \qquad \textbf{(2.16c)}$$

$$h_4(s) = -c_4\left((1\text{-}s) - \frac{\sinh[\sigma(1\text{-}s)]}{\sinh[\sigma]}\right) - c_3\left(s - \frac{\sinh[\sigma s]}{\sinh[\sigma]}\right) \qquad \textbf{(2.16d)}$$

where

$$c_1 = 1 - c_2 \qquad \textbf{(2.17a)}$$

$$c_2 = \frac{\sinh[\sigma]}{2\sinh[\sigma] - \sigma\cosh[\sigma] - \sigma} \qquad \textbf{(2.17b)}$$

$$c_3 = \frac{-\alpha}{\left(\beta^2 - \alpha^2\right)}\sinh[\sigma] \qquad \textbf{(2.17c)}$$

$$c_4 = \frac{\beta}{\left(\beta^2 - \alpha^2\right)}\sinh[\sigma] \qquad \textbf{(2.17d)}$$

$$\alpha = \sigma\cosh[\sigma] - \sinh[\sigma] \qquad \textbf{(2.17e)}$$

$$\beta = \sinh[\sigma] - \sigma \qquad \textbf{(2.17f)}$$

Equations (2.15) to (2.17) can be used to interpolate any function on an interval [0,1], when the function's values and its first derivatives are known at the end points of the interval. The application of these equations to algebraic grid generation is straightforward. Consider, for example, the two-boundary technique (ref. 1). Suppose we want to generate a grid system between two constant-η boundary surfaces. The formulation in this case can be written as follows:

$$\begin{aligned} \mathbf{r}(\xi,\eta,\zeta) &= \mathbf{r}(\xi,\eta=0,\zeta)\,h_1(\eta) + \mathbf{r}(\xi,\eta=1,\zeta)\,h_2(\eta) \\ &+ \frac{\partial \mathbf{r}(\xi,\eta=0,\zeta)}{\partial\eta}\,h_3(\eta) + \frac{\partial \mathbf{r}(\xi,\eta=1,\zeta)}{\partial\eta}\,h_4(\eta) \end{aligned} \qquad \textbf{(2.18)}$$

where

$$\mathbf{r}(\xi,\eta,\zeta) = \begin{bmatrix} x(\xi,\eta,\zeta) \\ y(\xi,\eta,\zeta) \\ z(\xi,\eta,\zeta) \end{bmatrix} \tag{2.19}$$

The above equation has the same form as if Hermite interpolation were used, except for the definition of the functions h_1, h_2, h_3, and h_4, which in the case of Hermite interpolation are cubic polynomials (ref. 1). Note, however, that the functions used in tension-spline interpolation (eqs. (2.16) and (2.17)) have the property that as $\sigma \to 0$, h_1, h_2, h_3, and h_4 approach the cubic polynomials used in Hermite interpolation (see ref. 1). Also, as $\sigma \to \infty$, $h_1(s) \to (1-s)$, $h_2(s) \to s$, $h_3(s) \to 0$, and $h_4(s) \to 0$, giving rise to linear connecting functions (Lagrange interpolation; see ref. 1).

2.3 Specification of Derivatives at Boundaries

When the two- and four-boundary methods, as described in reference 1, are used to generate grid systems, the first-order derivatives involved (e.g., $\partial \mathbf{r}(\xi,\eta = 0,\zeta)/\partial\eta$ and $\partial \mathbf{r}(\xi,\eta = 1,\zeta)/\partial\eta$ in equation (2.18)) need to be specified. In GRID3D-v1 and GRID3D-v2, these derivatives are specified such that grid lines intersect boundary surfaces orthogonally. In this section, the methods used in GRID3D-v1 and GRID3D-v2 to calculate these derivatives will be explained. The two-boundary technique given by equation (2.18) will be used to illustrate the concepts.

In GRID3D-v1, the derivatives $\partial \mathbf{r}(\xi,\eta = 0,\zeta)/\partial\eta$ and $\partial \mathbf{r}(\xi,\eta = 1,\zeta)/\partial\eta$ (eq. (2.18)) are chosen as follows:

$$\frac{\partial \mathbf{r}(\xi,\eta = 0,\zeta)}{\partial\eta} = K_{\eta 1}(\xi,\zeta)\,\mathbf{t}_{\eta 1} \qquad \frac{\partial \mathbf{r}(\xi,\eta = 1,\zeta)}{\partial\eta} = K_{\eta 2}(\xi,\zeta)\,\mathbf{t}_{\eta 2} \tag{2.20}$$

where

$$\mathbf{t}_{\eta 1} = -\frac{\partial \mathbf{r}(\xi,\eta = 0,\zeta)}{\partial\xi} \times \frac{\partial \mathbf{r}(\xi,\eta = 0,\zeta)}{\partial\zeta} \tag{2.21a}$$

$$\mathbf{t}_{\eta 2} = -\frac{\partial \mathbf{r}(\xi,\eta = 1,\zeta)}{\partial\xi} \times \frac{\partial \mathbf{r}(\xi,\eta = 1,\zeta)}{\partial\zeta} \tag{2.21b}$$

In equation (2.20), the terms $K_{\eta 1}(\xi,\zeta)$ and $K_{\eta 2}(\xi,\zeta)$ -- that is,the K-factors -- are specified by the user and are intended to control the magnitude of the derivatives. However, the magnitude of the

derivatives will also depend on the magnitude of the vectors $\mathbf{t}_{\eta 1}$ and $\mathbf{t}_{\eta 2}$, which in turn depend both on the geometry of the boundary surfaces at $\eta = 0$ and $\eta = 1$, respectively, and on the grid spacing on the surfaces. Thus, full control cannot be exerted over the magnitudes of the derivative terms $\partial\mathbf{r}(\xi,\eta=0,\zeta)/\partial\eta$ and $\partial\mathbf{r}(\xi,\eta=1,\zeta)/\partial\eta$, and for complex-shaped geometries this may pose a problem.

In order to overcome the aforementioned problems, in GRID3D-v2 the derivative terms $\partial\mathbf{r}(\xi,\eta = 0,\zeta)/\partial\eta$ and $\partial\mathbf{r}(\xi,\eta = 1,\zeta)/\partial\eta$ were defined as follows:

$$\frac{\partial\mathbf{r}(\xi,\eta = 0,\zeta)}{\partial\eta} = K_{\eta 1}(\xi,\zeta)\,\mathbf{e}_{\eta 1} \qquad \frac{\partial\mathbf{r}(\xi,\eta = 1,\zeta)}{\partial\eta} = K_{\eta 2}(\xi,\zeta)\,\mathbf{e}_{\eta 2} \qquad \textbf{(2.22)}$$

where $\mathbf{e}_{\eta 1}$ and $\mathbf{e}_{\eta 2}$ are unit vectors that are normal to the boundary surfaces at $\eta = 0$ and $\eta = 1$, respectively; that is

$$\mathbf{e}_{\eta 1} = \frac{\mathbf{t}_{\eta 1}}{\left|\mathbf{t}_{\eta 1}\right|} \qquad \mathbf{e}_{\eta 2} = \frac{\mathbf{t}_{\eta 2}}{\left|\mathbf{t}_{\eta 2}\right|} \qquad \textbf{(2.23)}$$

This approach allows total control over the magnitude of the derivatives $\partial\mathbf{r}(\xi,\eta = 0,\zeta)/\partial\eta$ and $\partial\mathbf{r}(\xi,\eta = 1,\zeta)/\partial\eta$ through the K-factors alone.

Finally, note that in GRID3D-v1, the K-factors were taken to be constants on each boundary surface, even though the authors realized that they could be allowed to vary. In GRID3D-v2, the K-factors are allowed to vary from point to point and can be controlled by the user.

In the next section, we show how GRID3D-v2 is used to generate grid systems. Several auxiliary programs were written to assist grid generation with GRID3D-v2. A description of these programs is given in appendix A. A complete listing of the GRID3D-v2 computer program is given in appendix B.

3.0 USING GRID3D-v2

In this section, the use of GRID3D-v2 is described. The section consists of two parts: first, an explanation of generating 3-D grid system with GRID3D-v2; and second, an example of such a grid system generated by using GRID3D-v2.

3.1 Generating a Three-Dimensional Grid System with GRID3D-v2

When using GRID3D-v2, the user must answer the following questions:

(1) Should the two boundary technique or the four boundary technique be used?
(2) Should any edge curve be defined by specifying the grid points directly?
(3) How many grid points are desired in each of the ξ, η, and ζ directions?
(4) Is any clustering of grid points needed?
(5) What K-factors should be used (see section 2.3 and pp. 18 to 20 in ref. 1) and are constant K-factors sufficient?

Once these questions have been answered, an input file for GRID3D-v2 must be constructed. The form of the input file is shown in figure 3-1. The various parameters in the input file are explained in table 3.1. The surface and edge curve numbering scheme embedded in GRID3D-v2 is shown in figure 3-2. Note that some of the edge curves are identical; that is, curves 3 and 9 are identical and so are curves 4 and 13, curves 7 and 10, and curves 8 and 14 (for further explanation of the edge curve numbering scheme, see p. 5 of ref. 2)

In the input file are values for K-factors at boundary surfaces -- a single value is assigned for all grid points on each surface. The user can modify the K-factor for any individual grid point or groups of grid points on each surface by adding FORTRAN statements into subroutine KFACTOR (see listing in Appendix B). Note that in addition to being used to generate grid points in the interior of the spatial domain, K-factors are also used to generate grid points on boundary surfaces themselves. For this latter case, the relevant K-factors are those at grid points that lie on the four edge curves bounding the surface. These K-factors can also be modified in subroutine KFACTOR.

Once an input file has been prepared (and an executable file created for GRID3D-v2 if subroutine KFACTOR was modified), GRID3D-v2 can be executed. Note, GRID3D-v2 reads the input file from unit 7 and writes output to unit 8.

Grid generation is an iterative process; that is, an acceptable grid system is generated by trial and error after generating a series of unacceptable grid systems. Some observations and rules of thumb that might be useful when generating grid systems with GRID3D are as follows:

(1) If it is necessary to partition a spatial domain, then select partitioning surfaces that intersect boundary surfaces as orthogonally as possible. This approach minimizes skewness both at boundaries and in the interior of the domain.

(2) A common boundary between two partitions should be specified in exactly the same way in the input files in order to guarantee a continuous grid across the common boundary.

(3) For improved flexibility in modifying a grid system, the boundary curves should be defined by using node points for tension-spline interpolation and a stretching function (see Section 2.1), rather than specifying grid points directly.

(4) When generating a grid system for a complicated geometry, first find stretching functions and K-factors for edge curves that give the desired distribution of grid points on the boundary surfaces. Afterwards, try to optimize grid-point distribution in the interior by modifying K-factors on boundary surfaces and/or the amount of tension in the connecting curves. If this two-step process does not yield an acceptable grid system, then try to modify grid-point distribution on boundary surfaces and/or re-partition the domain.

(5) Start with low tension and low K-factors. Slowly increase K-factors to improve orthogonality at boundaries and eliminate overlapping grid lines. Increasing the amount of tension in the connecting curves also can eliminate overlapping grid lines by straightening grid lines.

(6) Increased tension tends to straighten out grid lines whereas increased K-factors tend to increase the curvature of grid lines.

(7) If K-factors are too high, then grid lines can overlap.

(8) K-factors affect the grid spacing near the boundary surfaces. Increasing the K-factor at a boundary increases the grid spacing adjacent to that boundary. This effect can be beneficial in some instances but detrimental in others.

(9) K-factors should vary smoothly from grid line to grid line. An exception to this general rule is when a grid line intersects a boundary surface at a cusp in the surface.

3.2 Example: A Spatial Domain With Irregular Boundaries

Figure 1-1 shows a cooling passage in a radial turbine blade. Figure 3-3 shows how this cooling passage was partitioned into blocks or zones for the purpose of grid generation. The partitioning that is shown was deemed necessary in order to get an acceptable grid system.

Figure 3-4 shows the grid system for partition number 18. The input file for this partition is given in table 3-2. The entire grid system generated by using GRID3D-v2 is shown in figures 3-5 and 3-6. The plotting in figures 3-4 to 3-6 were obtained by using 3DSURF -- the plotting package supplied with GRID2D/3D.

4.0 SUMMARY

A new version of the grid generation program GRID3D, which is a part of the grid generation package GRID2D/3D, has been developed. The new program is referred to as GRID3D-v2. This report describes GRID3D-v2 and how to use it. The capability of the program was demonstrated by generating a grid system for a very complicated geometry, namely a cooling passage inside a radial turbine blade.

Appendix A -- Support Programs

A.1 Description of Support Programs

To support the task of grid generation with GRID3D-v2, several auxiliary programs have been developed. Two of these programs, namely PRSURF and 3DSURF, were developed to allow the user to view grid systems generated with the package. Three more programs, namely 3DPREP, EDGE, and EDGPREP, were developed to aid in the preparation of input files for GRID3D-v2. Finally, one program, GRIDTST, was written to test the grid system for problems such as overlapping grid lines. Each of these programs and their use will be described in the following pages.

3DSURF

The program 3DSURF was developed alongside the original version of GRID2D/3D to allow the user to plot grid systems on the computer screen. It was designed for IBM PC, XT and AT computer systems and compatibles. Two-dimensional grid output files from GRID2D/3D are already in the proper format for use with 3DSURF. Three-dimensional grid output files must, on the other hand, be processed using PRSURF (described next) to create input files for 3DSURF. A general description of 3DSURF and its use is given in Section 3.1 in reference 2.

PRSURF

The PRSURF program was written to process output files from GRID3D and create input files for 3DSURF. This program allows the user to select surfaces or parts of surfaces from the grid system (i.e., constant-ξ, constant-η, and constant-ζ surfaces) and to store them in a format compatible with 3DSURF. The program is interactive and self-explanatory.

PRSURF uses three files. The file containing the input to PRSURF (i.e., the output file from GRID3D) is read from unit 7. The output file from PRSURF (which becomes the input file for 3DSURF) is written to unit 9. Last, a file used for temporary storage of data is accessed as unit 8.

Finally, we mention that since 3DSURF is limited to handling surfaces that have 40 grid points per side or less, PRSURF automatically breaks any grid surface into sections that are 40 grid points by 40 grid points or smaller.

3DPREP

The user can apply 3DPREP to create input files for GRID3D-v2. The program prompts the user for all control parameters such as stretching functions and k-factors. It reads from files the coordinates of points defining the edge curves. These files must have the following format:

NP (number of points)

x_1, y_1, z_1

x_2, y_2, z_2

x_3, y_3, z_3

.

.

.

x_{NP}, y_{NP}, z_{NP}

The program can read data points from these files in both forward and reverse order, and the user can let the program read data from several files (the whole file or only a part of the file) to put together a single edge curve. 3DPREP is useful, primarily, when many input files for GRID3D-v2 need to be created from the same set of data.

3DPREP is designed to run on the IBM PC, XT, AT, and compatibles, but it can easily be modified to run on other computer systems. The only modification that should be needed in such a case is the insertion of open statements into the program so that the user can interactively specify which files the program must access.

EDGE

The EDGE program was written to aid the user in generating grid points along an edge curve that cannot be represented by a single spline curve (e.g., edge curves possessing derivative discontinuities such as cusps) and which must, therefore, be defined in the input files for GRID3D-v2 by giving the grid point coordinates directly (see Section 2.1). The program was designed to

generate grid points on an edge curve that is composed of several sections, where each section is defined by a set of nodal points that are interpolated by a spline curve. The number of grid points on each section and their distribution within the section is controlled independently.

EDGE, which is designed to run on the IBM PC, XT, AT, and compatibles, reads input data from UNIT 1 and writes the output to UNIT 20. The input file must have the following format:

NS (number of sections)

Data for section 1

Data for section 2

⋮

Data for section NS

where the data for an arbitrary section number i are the following:

IP_i (number of grid points on section i)
σ_i (tension parameter for the spline interpolation)
NN_i (number of node points given on section i)
x_1, y_1, z_1
x_2, y_2, z_2
x_3, y_3, z_3
.
.
.
$x_{NN_i}, y_{NN_i}, z_{NN_i}$
$StretchType_i$
$Beta1_i, Beta2_i$

The meaning of the parameters StretchType, Beta1, and Beta2 is explained in table 3-1. Note that in EDGE it is assumed that the curve being generated is continuous; that is, the last grid point on one

section must be the same as the first grid point on the next section. Thus the output from EDGE has the following format:

IL (number of grid points on the edge curve)

$x_{1,1}, y_{1,1}, z_{1,1}$
$x_{1,2}, y_{1,2}, z_{1,2}$
$x_{1,3}, y_{1,3}, z_{1,3}$
.
.
.
$x_{1,IP_1-1}, y_{1,IP_1-1}, z_{1,IP_1-1}$
$x_{2,1}, y_{2,1}, z_{2,1}$
$x_{2,2}, y_{2,2}, z_{2,2}$
$x_{2,3}, y_{2,3}, z_{2,3}$
.
.
.
$x_{2,IP_2-1}, y_{2,IP_2-1}, z_{2,IP_2-1}$
.
.
.
$x_{NS,1}, y_{NS,1}, z_{NS,1}$
$x_{NS,2}, y_{NS,2}, z_{NS,2}$
$x_{NS,3}, y_{NS,3}, z_{NS,3}$
.
.
.
$x_{NS,IP_{NS}}, y_{NS,IP_{NS}}, z_{NS,IP_{NS}}$

where $x_{i,j}$, $y_{i,j}$, and $z_{i,j}$ are the coordinates of grid point number j on section number i. The total number of grid points is

$$IL = 1 + \sum_{i=1}^{NS} (IP_i - 1)$$

Intentionally, this is the same format as for the input files for 3DPREP so the output files from EDGE can be read by 3DPREP.

GRIDTST

The GRIDTST program is used to check grid systems of 3-D spatial domains for defects, such as over lapping grid lines, that result in a negative Jacobian, where the Jacobian is defined as follows (for further explanation of the Jacobian, see ref. 1):

$$J = x_\xi(y_\eta z_\zeta - y_\zeta z_\eta) - x_\eta(y_\xi z_\zeta - y_\zeta z_\xi) + x_\zeta(y_\xi z_\eta - y_\eta z_\xi)$$

GRIDTST evaluates the Jacobian at every grid point, estimating the derivatives (i.e., x_ξ, y_ξ, z_ξ, etc.) by using central difference approximations for grid points that do not lie on the boundary surfaces of the domain and by using second order accurate one-sided difference formulas where necessary on the boundary surfaces. If any negative Jacobians are found, GRIDTST prints a message on the computer screen, and the Jacobians and the grid point locations are written into a file.

The input into GRIDTST is the grid system generated by GRID3D-v2. The input is read from UNIT 1 whereas the output (if any) is written into UNIT 2. GRIDTST is written to run on the IBM PC, AT, XT, and compatibles, but it can be used on any computer system.

A.2 Listing of PRSURF

```
      PROGRAM PRSURF
C
C======================================================================
C   This program writes out user picked surfaces from a 3-D grid     I
C======================================================================
C
       PARAMETER (IM=11,JM=51,KM=151)

       INTEGER  i, j, k, IL, JL, KL

       REAL  X(IM,JM,KM),Y(IM,JM,KM),Z(IM,JM,KM)
C
       IERR=0
5      WRITE(*,5001)
       READ(*,*)IPICK
5001   FORMAT('2',///,' Please enter:',//,
     $'       1 - if you want the boundary surfaces of the grid',/,
     $'           to be saved',/,
     $'       2 - if you want to select surfaces to be saved',///)
C
       IF(IPICK.EQ.1)THEN
          CALL PRGRID(X,Y,Z,IM,JM,KM)
       ELSE IF(IPICK.EQ.2)THEN
          CALL PRSRFS(X,Y,Z,IM,JM,KM)
       ELSE IF(IERR.EQ.0)THEN
          WRITE(*,*)'     You will get one more chance to make a'
          WRITE(*,*)'     selection - Please enter any character'
          WRITE(*,*)'           and then press  RETURN'
          READ(*,*)
          IERR=1
          GOTO 5
       ELSE
          STOP
       ENDIF
C
C
       STOP
       END
C
C
C
C
       SUBROUTINE PRGRID(XPnt,YPnt,ZPnt,IL2,JL2,KL2)

C This subroutine reads in grid point coordinates and writes out the
C coordinates of the grid points which lie along specified planes.


       INTEGER  i, j, k, IL, JL, KL, IL1, JL1, KL1

       REAL  XPnt(IL2,JL2,KL2),
     $       YPnt(IL2,JL2,KL2),
     $       ZPnt(IL2,JL2,KL2)
```

```
C Read in the grid size.

      READ(7,*) IL
      READ(7,*) JL
      READ(7,*) KL

C Read in the grid point locations.

      DO 7 i=1,IL
         DO 6 j=1,JL
            DO 5 k=1,KL
               READ(7,*) XPnt(i,j,k),YPnt(i,j,k),ZPnt(i,j,k)
 5          CONTINUE
 6       CONTINUE
 7    CONTINUE

C Calculate the number of sections the grid must be split into
C for plotting purposes.  (Plotting routines can handle only a grid
C with a maximum dimension of 40.  Here it is assumed that only the
C zeta-coordinate direction can involve more grid points than that)

      KS=(KL-1)/39
      JS=(JL-1)/39

C Print out the number of surfaces.

      IF((KS*39+1).LT.KL)THEN
         NSK=KS+1
      ELSE
         NSK=KS
      ENDIF
      IF((JS*39+1).LT.JL)THEN
         NSJ=JS+1
      ELSE
         NSJ=JS
      ENDIF

      NOSURF=2*NSJ+2*NSK+3*(NSK*NSJ)

      WRITE(9,*) NOSURF

C Print out the grid points.

      DO 21 m=1,NSJ

      j0=39*(m-1)+1
      j1=MIN(j0+39,JL)
      JLm=j1-j0+1

      WRITE(9,*) IL
      WRITE(9,*) JLm

      DO 20 i=1,IL
         DO 10 j=j0,j1
            WRITE(9,25) XPnt(i,j,KL),YPnt(i,j,KL),ZPnt(i,j,KL)
 10      CONTINUE
 20   CONTINUE
```

```
21    CONTINUE

25    FORMAT(1X,F10.6,3x,F10.6,3X,F10.6)

      DO 29 n=1,NSK

      k0=39*(n-1)+1
      k1=MIN(k0+39,KL)
      KLn=k1-k0+1

      WRITE(9,*) IL
      WRITE(9,*) KLN

      DO 28 i=1,IL
         DO 27 k=k0,k1
            WRITE(9,25) XPnt(i,JL,k),YPnt(i,JL,k),ZPnt(i,JL,k)
27       CONTINUE
28    CONTINUE
29    CONTINUE


      DO 44 m=1,NSJ

      j0=39*(m-1)+1
      j1=MIN(j0+39,JL)
      JLm=j1-j0+1

      DO 44 n=1,NSK

      k0=39*(n-1)+1
      k1=MIN(k0+39,KL)
      KLn=k1-k0+1

      WRITE(9,*) JLm
      WRITE(9,*) KLn

      DO 43 j=j0,j1
         DO 42 k=k0,k1
            WRITE(9,25) XPnt(IL,j,k),YPnt(IL,j,k),ZPnt(IL,j,k)
42       CONTINUE
43    CONTINUE
44    CONTINUE


      DO 110 m=1,NSJ

      j0=39*(m-1)+1
      j1=MIN(j0+39,JL)
      JLm=j1-j0+1

      DO 110 n=1,NSK

      k0=39*(n-1)+1
      k1=MIN(k0+39,KL)
      KLn=k1-k0+1

      WRITE(9,*) JLm
      WRITE(9,*) KLn
```

```
      DO 109 j=j0,j1
         DO 108 k=k0,k1
            WRITE(9,25) XPnt(1,j,k),YPnt(1,j,k),ZPnt(1,j,k)
108      CONTINUE
109   CONTINUE
110   CONTINUE


      DO 131 m=1,NSJ

      j0=39*(m-1)+1
      j1=MIN(j0+39,JL)
      JLm=j1-j0+1

      WRITE(9,*) IL
      WRITE(9,*) JLm

      DO 130 i=1,IL
         DO 125 j=j0,j1
            WRITE(9,25) XPnt(i,j,1),YPnt(i,j,1),ZPnt(i,j,1)
125      CONTINUE
130   CONTINUE
131   CONTINUE

      DO 134 n=1,NSK

      k0=39*(n-1)+1
      k1=MIN(k0+39,KL)
      KLn=k1-k0+1

      WRITE(9,*) IL
      WRITE(9,*) KLn
      DO 133 i=1,IL
         DO 132 k=k0,k1
            WRITE(9,25) XPnt(i,1,k),YPnt(i,1,k),ZPnt(i,1,k)
132      CONTINUE
133   CONTINUE
134   CONTINUE


      DO 144 m=1,NSJ

      j0=39*(m-1)+1
      j1=MIN(j0+39,JL)
      JLm=j1-j0+1

      DO 144 n=1,NSK

      k0=39*(n-1)+1
      k1=MIN(k0+39,KL)
      KLn=k1-k0+1

      WRITE(9,*) JLm
      WRITE(9,*) KLn

      IH=(IL+1)/2
```

```
      DO 143 j=j0,j1
         DO 142 k=k0,k1
            WRITE(9,25) XPnt(IH,j,k),YPnt(IH,j,k),ZPnt(IH,j,k)
 142     CONTINUE
 143  CONTINUE
 144  CONTINUE


      STOP
      END
C
C
      SUBROUTINE PRSRFS(X,Y,Z,IM,JM,KM)

C This subroutine reads in grid point coordinates and writes out the     e
C coordinates of the grid points which lie along planes specified
C by the user.

      INTEGER  i, j, k, IL, JL, KL

      REAL  X(IM,JM,KM),Y(IM,JM,KM),Z(IM,JM,KM)
C
      WRITE(*,5001)
C
C Read in the grid size.

      READ(7,*) IL
      READ(7,*) JL
      READ(7,*) KL

C Read in the grid point locations.

      DO 3 i=1,IL
         DO 2 j=1,JL
            DO 1 k=1,KL
               READ(7,*) X(i,j,k),Y(i,j,k),Z(i,j,k)
 1          CONTINUE
 2       CONTINUE
 3    CONTINUE

      NOSURF=0
 5    WRITE(*,5002)IL,JL,KL
      READ(*,*)IPICK
      IF(IPICK.EQ.1)THEN
        WRITE(*,5003)
        READ(*,*)IPICK2
        IF(IPICK2.EQ.1)THEN
          WRITE(*,*)
          WRITE(*,*)'Please enter the value of i'
          READ(*,*)I
          JFIRST=1
          JLAST=JL
          KFIRST=1
          KLAST=KL
        ELSE
          WRITE(*,*)
          WRITE(*,*)'Please enter the value of i'
          READ(*,*)I
```

```
            WRITE(*,*)
            WRITE(*,*)'   Please enter the lower and upper limit'
            WRITE(*,*)'   for the j coordinate (JFIRST,JLAST)'
            READ(*,*)JFIRST,JLAST
            WRITE(*,*)
            WRITE(*,*)'   Please enter the lower and upper limit'
            WRITE(*,*)'   for the k coordinate (KFIRST,KLAST)'
            READ(*,*)KFIRST,KLAST
          ENDIF
          NSJ=(JLAST-JFIRST)/39
          IF(JFIRST+NSJ*39.LT.JLAST)NSJ=NSJ+1
          NSK=(KLAST-KFIRST)/39
          IF(KFIRST+NSK*39.LT.KLAST)NSK=NSK+1
          NOSURF=NOSURF+NSJ*NSK
          DO 60 m=1,NSJ
             j0=39*(m-1)+JFIRST
             j1=MIN(j0+39,JLAST)
             JLm=j1-j0+1
             DO 50 n=1,NSK
                k0=39*(n-1)+KFIRST
                k1=MIN(k0+39,KLAST)
                KLn=k1-k0+1
                WRITE(8,*) JLm
                WRITE(8,*) KLn
                DO 40 j=j0,j1
                   DO 30 k=k0,k1
                      WRITE(8,25) X(I,j,k),Y(I,j,k),Z(I,j,k)
 30                CONTINUE
 40             CONTINUE
 50          CONTINUE
 60       CONTINUE
          IERR=0
          GOTO 5
C
        ELSEIF(IPICK.EQ.2)THEN
          WRITE(*,5003)
          READ(*,*)IPICK2
          IF(IPICK2.EQ.1)THEN
            WRITE(*,*)
            WRITE(*,*)'Please enter the value of j'
            READ(*,*)J
            IFIRST=1
            ILAST=IL
            KFIRST=1
            KLAST=KL
          ELSE
            WRITE(*,*)
            WRITE(*,*)'Please enter the value of j'
            READ(*,*)J
            WRITE(*,*)
            WRITE(*,*)'   Please enter the lower and upper limit'
            WRITE(*,*)'   for the i coordinate (IFIRST,ILAST)'
            READ(*,*)IFIRST,ILAST
            WRITE(*,*)
            WRITE(*,*)'   Please enter the lower and upper limit'
            WRITE(*,*)'   for the k coordinate (KFIRST,KLAST)'
            READ(*,*)KFIRST,KLAST
          ENDIF
```

```
      NSI=(ILAST-IFIRST)/39
      IF(IFIRST+NSI*39.LT.ILAST)NSI=NSI+1
      NSK=(KLAST-KFIRST)/39
      IF(KFIRST+NSK*39.LT.KLAST)NSK=NSK+1
      NOSURF=NOSURF+NSI*NSK
      DO 100 m=1,NSI
         i0=39*(m-1)+IFIRST
         i1=MIN(j0+39,ILAST)
         ILm=i1-i0+1
         DO 90 n=1,NSK
            k0=39*(n-1)+KFIRST
            k1=MIN(k0+39,KLAST)
            KLn=k1-k0+1

            WRITE(8,*) ILm
            WRITE(8,*) KLn

            DO 80 i=i0,i1
               DO 70 k=k0,k1
                  WRITE(8,25) X(i,J,k),Y(i,J,k),Z(i,J,k)
 70            CONTINUE
 80         CONTINUE
 90      CONTINUE
 100  CONTINUE
      IERR=0
      GOTO 5
C
    ELSEIF(IPICK.EQ.3)THEN
      WRITE(*,5003)
      READ(*,*)IPICK2
      IF(IPICK2.EQ.1)THEN
        WRITE(*,*)
        WRITE(*,*)'Please enter the value of k'
        READ(*,*)K
        IFIRST=1
        ILAST=IL
        JFIRST=1
        JLAST=JL
      ELSE
        WRITE(*,*)
        WRITE(*,*)'Please enter the value of k'
        READ(*,*)K
        WRITE(*,*)
        WRITE(*,*)'   Please enter the lower and upper limit'
        WRITE(*,*)'   for the i coordinate (IFIRST,ILAST)'
        READ(*,*)IFIRST,ILAST
        WRITE(*,*)
        WRITE(*,*)'   Please enter the lower and upper limit'
        WRITE(*,*)'   for the j coordinate (JFIRST,JLAST)'
        READ(*,*)JFIRST,JLAST
      ENDIF
      NSI=(ILAST-IFIRST)/39
      IF(IFIRST+NSI*39.LT.ILAST)NSI=NSI+1
      NSJ=(JLAST-JFIRST)/39
      IF(JFIRST+NSJ*39.LT.JLAST)NSJ=NSJ+1
      NOSURF=NOSURF+NSI*NSJ
      DO 140 m=1,NSI
         i0=39*(m-1)+IFIRST
```

```
         i1=MIN(j0+39,ILAST)
         ILm=i1-i0+1
         DO 130 n=1,NSJ
            j0=39*(n-1)+JFIRST
            j1=MIN(j0+39,JLAST)
            JLn=j1-j0+1
            WRITE(8,*) ILm
            WRITE(8,*) JLn
            DO 120 i=i0,i1
               DO 110 j=j0,j1
                  WRITE(8,25) X(i,j,K),Y(i,j,K),Z(i,j,K)
110            CONTINUE
120         CONTINUE
130      CONTINUE
140   CONTINUE
      IERR=0
      GOTO 5
    ELSEIF(IPICK.EQ.0 .OR. IERR.EQ.1)THEN
      REWIND(8)
      WRITE(9,*)NOSURF
      DO 1000 N=1,NOSURF
         READ(8,*)I1
         READ(8,*)I2
         WRITE(9,*)I1
         WRITE(9,*)I2
         DO 999 I=1,I1*I2
            READ(8,*)X1,X2,X3
            WRITE(9,25)X1,X2,X3
999      CONTINUE
1000  CONTINUE
    ELSEIF(IERR.EQ.0)THEN
      IERR=1
      WRITE(*,*)'                 INVALID SELECTION'
      WRITE(*,*)'      You will get one more chance to make a'
      WRITE(*,*)'      selection - Please enter any character'
      WRITE(*,*)'            and then press  RETURN'
      READ(*,*)
      GOTO 5
    ENDIF
C
C
      RETURN
25    FORMAT(1X,F10.6,3x,F10.6,3X,F10.6)
5001  FORMAT(' ',///,
     $'        Reading in the grid. Please wait',///)
5002  FORMAT(' ',///,' Please enter:',//,
     $'        1 - if you want to save a constant-i surface',/,
     $'        2 - if you want to save a constant-j surface',/,
     $'        3 - if you want to save a constant-k surface',/,
     $'        0 - if you want to QUIT',/,
     $'(Recall: IL=',I3,',  JL=',I3,',  KL=',I3,')',//)
5003  FORMAT(' ',///,' Please enter:',//,
     $'        1 - if you want the whole surface saved',/,
     $'        2 - if you want to specify a part of the',/,
     $'            surface to be saved',//)
      END
```

A.3 Listing of 3DPREP

```
      PROGRAM PREPARE
C
       DIMENSION TENSION(16),BETA1(16),BETA2(16),
     /          X1(100),Y1(100),Z1(100),X(16,100),Y(16,100),Z(16,100)
       INTEGER STRTYPE(16),N2B(4),TYPE(16),NODES(16)
       REAL    KXI1,KXI2,KETA1,KETA2,KZETA1,KZETA2
C
       WRITE(*,*)
       WRITE(*,*)
       WRITE(*,*)
       WRITE(*,*)
       WRITE(*,*)' This program prepares input files for GRID3D by'
       WRITE(*,*)' reading the necessary information from the screen'
       WRITE(*,*)' and from files.'
       WRITE(*,*)
       WRITE(*,*)
C
       WRITE(*,*)'What technique is to be used.'
       WRITE(*,*)'Enter 2 for the two-boundary technique'
       WRITE(*,*)'  or  4 for the four-boundary technique'
       READ(*,*)ITECH
C
       WRITE(*,*)'Enter IL, JL and KL'
       READ(*,*)IL,JL,KL
C
       WRITE(*,*)'Enter SigmaXi, SigmaEta, and SigmaZeta'
       READ(*,*)SigXi,SigEt,SigZt
C
       WRITE(*,*)'Enter kXI1 and kXI2'
       READ(*,*)KXI1,KXI2
       WRITE(*,*)'Enter kETA1 and kETA2'
       READ(*,*)KETA1,KETA2
       WRITE(*,*)'Enter kZETA1 and kZETA2'
       READ(*,*)KZETA1,KZETA2
C
       WRITE(*,*)'The output file will be UNIT 20'
       WRITE(20,*)ITECH,'    Technique'
       WRITE(20,*)IL,'    IL'
       WRITE(20,*)JL,'    JL'
       WRITE(20,*)KL,'    KL'
       WRITE(20,*)SigXi,'    SigmaXi'
       WRITE(20,*)SigEt,'    SigmaEta'
       WRITE(20,*)SigZt,'    SigmaZeta'
       WRITE(20,*)KXI1,'    kXI1'
       WRITE(20,*)KXI2,'    kXI2'
       WRITE(20,*)KETA1,'    kETA1'
       WRITE(20,*)KETA2,'    kETA2'
       WRITE(20,*)KZETA1,'    kZETA1'
       WRITE(20,*)KZETA2,'    kZETA2'
C
C
       DO 200 NSRF=1,2
          NE1=1+4*(NSRF-1)
```

```
      NE2=NE1+1
      NE3=NE1+2
      NE4=NE1+3
C
C
C  Get data for edge NE1:
C
      WRITE(*,2002)NE1
5     WRITE(*,2001)NE1
      READ(*,*)ITYPE
      IF(ITYPE.NE.1 .AND. ITYPE.NE.2)GOTO 5
C
      IF(ITYPE.EQ.1)THEN
        CALL GETGRP(X1,Y1,Z1,NOP)
        IF(NOP.NE.KL)WRITE(*,*)
     /      'WARNING --- NUMBER OF GRID POINTS INCONSISTENT -',
     /      ' EDGE',NE1
        WRITE(20,3001)ITYPE,NE1
        DO 10 K=1,KL
           WRITE(20,3004)X1(K),Y1(K),Z1(K),K
10      CONTINUE
        X11=X1(1)
        Y11=Y1(1)
        Z11=Z1(1)
        X1L=X1(KL)
        Y1L=Y1(KL)
        Z1L=Z1(KL)
C
      ELSEIF(ITYPE.EQ.2)THEN
C
        CALL GETNODES(X1,Y1,Z1,NOP)
C
        WRITE(*,2006)NE1
        READ(*,*)TENSN
        CALL GETSTR(NE1,ISTR1,BETA11,BETA21)
C
        WRITE(20,3001)ITYPE,NE1
        WRITE(20,3002)TENSN
        WRITE(20,3003)NOP
        DO 20 I=1,NOP
           WRITE(20,3004)X1(I),Y1(I),Z1(I),I
20      CONTINUE
        WRITE(20,3005)ISTR1
        IF(ISTR1.NE.4)WRITE(20,3006)BETA11
        IF(ISTR1.EQ.4)WRITE(20,3007)BETA11,BETA21
        X11=X1(1)
        Y11=Y1(1)
        Z11=Z1(1)
        X1L=X1(NOP)
        Y1L=Y1(NOP)
        Z1L=Z1(NOP)
C
      ENDIF
C
C
C  Get data for edge NE2:
C
      WRITE(*,2002)NE2
```

```
25        WRITE(*,2001)NE2
          READ(*,*)ITYPE
          IF(ITYPE.NE.1 .AND. ITYPE.NE.2)GOTO 25
C
          IF(ITYPE.EQ.1)THEN
            CALL GETGRP(X1,Y1,Z1,NOP)
            IF(NOP.NE.KL)WRITE(*,*)
     /         'WARNING --- NUMBER OF GRID POINTS INCONSISTENT -',
     /         ' EDGE',NE2
            WRITE(20,3001)ITYPE,NE2
            DO 30 K=1,KL
               WRITE(20,3004)X1(K),Y1(K),Z1(K),K
30          CONTINUE
            X21=X1(1)
            Y21=Y1(1)
            Z21=Z1(1)
            X2L=X1(KL)
            Y2L=Y1(KL)
            Z2L=Z1(KL)
C
          ELSEIF(ITYPE.EQ.2)THEN
C
            CALL GETNODES(X1,Y1,Z1,NOP)
C
            WRITE(*,2006)NE2
            READ(*,*)TENSN
            CALL GETSTR(NE1,ISTR2,BETA12,BETA22)
C
            WRITE(20,3001)ITYPE,NE2
            WRITE(20,3002)TENSN
            WRITE(20,3003)NOP
            DO 40 I=1,NOP
               WRITE(20,3004)X1(I),Y1(I),Z1(I),I
40          CONTINUE
            WRITE(20,3005)ISTR2
            IF(ISTR2.NE.4)WRITE(20,3006)BETA12
            IF(ISTR2.EQ.4)WRITE(20,3007)BETA12,BETA22
            X21=X1(1)
            Y21=Y1(1)
            Z21=Z1(1)
            X2L=X1(NOP)
            Y2L=Y1(NOP)
            Z2L=Z1(NOP)
C
          ENDIF
C
C
C  Get data for edge NE3:
C
          WRITE(*,2002)NE3
45        WRITE(*,2003)NE3
          READ(*,*)ITYPE
          IF(ITYPE.NE.1 .AND. ITYPE.NE.2 .AND. ITYPE.NE.3)GOTO 45
C
          IF(ITYPE.EQ.1)THEN
            CALL GETGRP(X1,Y1,Z1,NOP)
            IF(NOP.NE.IL)WRITE(*,*)
     /         'WARNING --- NUMBER OF GRID POINTS INCONSISTENT -',
```

```
     /          ' EDGE',NE3
             WRITE(20,3001)ITYPE,NE3
             DO 50 I=1,IL
                WRITE(20,3004)X1(I),Y1(I),Z1(I),I
50           CONTINUE
C
           ELSEIF(ITYPE.EQ.2)THEN
C
             CALL GETNODES(X1,Y1,Z1,NOP)
C
             WRITE(*,2006)NE3
             READ(*,*)TENSN
             CALL GETSTR(NE3,ISTR3,BETA13,BETA23)
C
             WRITE(20,3001)ITYPE,NE3
             WRITE(20,3002)TENSN
             WRITE(20,3003)NOP
             DO 60 I=1,NOP
                WRITE(20,3004)X1(I),Y1(I),Z1(I),I
60           CONTINUE
             WRITE(20,3005)ISTR3
             IF(ISTR3.NE.4)WRITE(20,3006)BETA13
             IF(ISTR3.EQ.4)WRITE(20,3007)BETA13,BETA23
C
           ELSEIF(ITYPE.EQ.3)THEN
             ITYPE=2
             NOP=2
             I1=1
             I2=2
             WRITE(*,2006)NE3
             READ(*,*)TENSN
             CALL GETSTR(NE3,ISTR3,BETA13,BETA23)
             WRITE(20,3001)ITYPE,NE3
             WRITE(20,3002)TENSN
             WRITE(20,3003)NOP
             WRITE(20,3004)X11,Y11,Z11,I1
             WRITE(20,3004)X21,Y21,Z21,I2
             X1(1)=X11
             X1(2)=X21
             Y1(1)=Y11
             Y1(2)=Y21
             Z1(1)=Z11
             Z1(2)=Z21
             WRITE(20,3005)ISTR3
             IF(ISTR3.NE.4)WRITE(20,3006)BETA13
             IF(ISTR3.EQ.4)WRITE(20,3007)BETA13,BETA23
C
           ENDIF
C
           TYPE(8+NSRF)=ITYPE
           NODES(8+NSRF)=NOP
           TENSION(8+NSRF)=TENSN3
           STRTYPE(8+NSRF)=ISTR3
           BETA1(8+NSRF)=BETA13
           BETA2(8+NSRF)=BETA23
           IF(ITYPE.EQ.1)IMAX=IL
           IF(ITYPE.EQ.2)IMAX=NOP
           DO 70 I=1,IMAX
```

```
            X(8+NSRF,I)=X1(I)
            Y(8+NSRF,I)=Y1(I)
            Z(8+NSRF,I)=Z1(I)
70        CONTINUE
C
C
C  Get data for edge NE4:
C
          WRITE(*,2002)NE4
75        WRITE(*,2003)NE4
          READ(*,*)ITYPE
          IF(ITYPE.NE.1 .AND. ITYPE.NE.2 .AND. ITYPE.NE.3)GOTO 75
C
          IF(ITYPE.EQ.1)THEN
            CALL GETGRP(X1,Y1,Z1,NOP)
            IF(NOP.NE.IL)WRITE(*,*)
     /         'WARNING --- NUMBER OF GRID POINTS INCONSISTENT -',
     /         ' EDGE',NE4
            WRITE(20,3001)ITYPE,NE4
            DO 80 I=1,IL
               WRITE(20,3004)X1(K),Y1(K),Z1(K),K
80          CONTINUE
C
          ELSEIF(ITYPE.EQ.2)THEN
C
            CALL GETNODES(X1,Y1,Z1,NOP)
C
            WRITE(*,2006)NE4
            READ(*,*)TENSN
            CALL GETSTR(NE4,ISTR4,BETA14,BETA24)
C
            WRITE(20,3001)ITYPE,NE4
            WRITE(20,3002)TENSN
            WRITE(20,3003)NOP
            DO 90 I=1,NOP
               WRITE(20,3004)X1(I),Y1(I),Z1(I),I
90          CONTINUE
            WRITE(20,3005)ISTR4
            IF(ISTR4.NE.4)WRITE(20,3006)BETA14
            IF(ISTR4.EQ.4)WRITE(20,3007)BETA14,BETA24
C
          ELSEIF(ITYPE.EQ.3)THEN
            ITYPE=2
            NOP=2
            I1=1
            I2=2
            WRITE(*,2006)NE4
            READ(*,*)TENSN
            CALL GETSTR(NE4,ISTR4,BETA14,BETA24)
            WRITE(20,3001)ITYPE,NE4
            WRITE(20,3002)TENSN
            WRITE(20,3003)NOP
            WRITE(20,3004)X1L,Y1L,Z1L,I1
            WRITE(20,3004)X2L,Y2L,Z2L,I2
            X1(1)=X1L
            X1(2)=X2L
            Y1(1)=Y1L
            Y1(2)=Y2L
```

```
          Z1(1)=Z1L
          Z1(2)=Z2L
          WRITE(20,3005)ISTR4
          IF(ISTR4.NE.4)WRITE(20,3006)BETA14
          IF(ISTR4.EQ.4)WRITE(20,3007)BETA14,BETA24
C
        ENDIF
C
        TYPE(12+NSRF)=ITYPE
        NODES(12+NSRF)=NOP
        TENSION(12+NSRF)=TENSN4
        STRTYPE(12+NSRF)=ISTR4
        BETA1(12+NSRF)=BETA14
        BETA2(12+NSRF)=BETA24
        IF(ITYPE.EQ.1)IMAX=IL
        IF(ITYPE.EQ.2)IMAX=NOP
        DO 100 I=1,IMAX
           X(16+NSRF,I)=X1(I)
           Y(16+NSRF,I)=Y1(I)
           Z(16+NSRF,I)=Z1(I)
100     CONTINUE
C
C
200   CONTINUE
C
C
      IF(ITECH.EQ.2)THEN
         WRITE(*,'(/////////)')
         WRITE(*,*)'   Specify stretching parameters for edges'
         WRITE(*,*)'   11, 12, 15 and 16'
         WRITE(*,'(////)')
         N2B(1)=11
         N2B(2)=12
         N2B(3)=15
         N2B(4)=16
         DO 201 IE=1,4
            CALL GETSTR(N2B(IE),ISTR,BETA1IE,BETA2IE)
            WRITE(20,3010)ISTR,N2B(IE)
            IF(ISTR.NE.4)WRITE(20,3006)BETA1IE
            IF(ISTR.EQ.4)WRITE(20,3007)BETA1IE,BETA2IE
201      CONTINUE
C
      ELSEIF(ITECH.EQ.4)THEN
C
         DO 500 NSRF=3,4
            NE1=1+4*(NSRF-1)
            NE2=NE1+1
            NE3=NE1+2
            NE4=NE1+3
C
C
C       Write data for edge NE1:
C
         WRITE(20,3001)TYPE(NE1),NE1
         IF(TYPE(NE1).EQ.1)THEN
           DO 210 I=1,IL
              WRITE(20,3004)X(NE1,I),Y(NE1,I),Z(NE1,I),I
210        CONTINUE
```

```
C
         ELSEIF(TYPE(NE1).EQ.2)THEN
C
           WRITE(20,3002)TENSION(NE1)
           WRITE(20,3003)NODES(NE1)
           DO 220 I=1,NODES(NE1)
              WRITE(20,3004)X(NE1,I),Y(NE1,I),Z(NE1,I),I
220        CONTINUE
           WRITE(20,3005)STRTYPE(NE1)
           IF(STRTYPE(NE1).NE.4)WRITE(20,3006)BETA1(NE1)
           IF(STRTYPE(NE1).EQ.4)
     &           WRITE(20,3007)BETA1(NE1),BETA2(NE1)
C
         ENDIF
C
C
C      Write data for edge NE2:
C
         WRITE(20,3001)TYPE(NE2),NE2
         IF(TYPE(NE2).EQ.1)THEN
           DO 230 I=1,IL
              WRITE(20,3004)X(NE2,I),Y(NE2,I),Z(NE2,I),I
230        CONTINUE
C
         ELSEIF(TYPE(NE2).EQ.2)THEN
C
           WRITE(20,3002)TENSION(NE2)
           WRITE(20,3003)NODES(NE2)
           DO 240 I=1,NODES(NE2)
              WRITE(20,3004)X(NE2,I),Y(NE2,I),Z(NE2,I),I
240        CONTINUE
           WRITE(20,3005)STRTYPE(NE2)
           IF(STRTYPE(NE1).NE.4)WRITE(20,3006)BETA1(NE2)
           IF(STRTYPE(NE1).EQ.4)
     &           WRITE(20,3007)BETA1(NE2),BETA2(NE2)
C
         ENDIF
C
C
C
C      Get data for edge NE3:
C
305         WRITE(*,2002)NE3
C
            WRITE(*,*)'Enter the following:'
            WRITE(*,*)'  1  if grid points are specified'
            WRITE(*,*)'  2  if nodes for splining are specified'
            WRITE(*,*)'  3  if you want to let the end points of'
            WRITE(*,*)'     edges already entered define the curve'
            WRITE(*,*)'  4  if you want to use the end points of'
            WRITE(*,*)'     edges already defined but add (by '
            WRITE(*,*)'     typing in directly) some points in '
            WRITE(*,*)'     between'
            WRITE(*,*)
            WRITE(*,*)
            WRITE(*,*)
            WRITE(*,*)'Enter your choice'
            READ(*,*)ITYPE
```

```
C
          IF(ITYPE.NE.1 .AND. ITYPE.NE.2 .AND. ITYPE.NE.3
     /                  .AND. ITYPE.NE.4)GOTO 305
C
C
          IF(ITYPE.EQ.1)THEN
            CALL GETGRP(X1,Y1,Z1,NOP)
            IF(NOP.NE.JL)WRITE(*,*)
     /       'WARNING --- NUMBER OF GRID POINTS INCONSISTENT -',
     /       ' EDGE',NE3
            WRITE(20,3001)ITYPE,NE3
            DO 310 J=1,JL
               WRITE(20,3004)X1(J),Y1(J),Z1(J),J
310          CONTINUE
C
C
          ELSEIF(ITYPE.EQ.2)THEN
            CALL GETNODES(X1,Y1,Z1,NOP)
C
            WRITE(*,2006)NE3
            READ(*,*)TENSN
            CALL GETSTR(NE3,ISTR3,BETA13,BETA23)
C
            WRITE(20,3001)ITYPE,NE3
            WRITE(20,3002)TENSN
            WRITE(20,3003)NOP
            DO 320 I=1,NOP
               WRITE(20,3004)X1(I),Y1(I),Z1(I),I
320          CONTINUE
            WRITE(20,3005)ISTR3
            IF(ISTR3.NE.4)WRITE(20,3006)BETA13
            IF(ISTR3.EQ.4)WRITE(20,3007)BETA13,BETA23
C
          ELSEIF(ITYPE.EQ.3)THEN
            ITYPE=2
            NOP=2
            I1=1
            I2=2
C
            WRITE(*,2006)NE3
            READ(*,*)TENSN
            CALL GETSTR(NE3,ISTR3,BETA13,BETA23)
C
            WRITE(20,3001)ITYPE,NE3
            WRITE(20,3002)TENSN
            WRITE(20,3003)NOP
            WRITE(20,3004)X(NE1,1),Y(NE1,1),Z(NE1,1),I1
            WRITE(20,3004)X(NE2,1),Y(NE2,1),Z(NE2,1),I2
            WRITE(20,3005)ISTR3
            IF(ISTR3.NE.4)WRITE(20,3006)BETA13
            IF(ISTR3.EQ.4)WRITE(20,3007)BETA13,BETA23
C
C
          ELSEIF(ITYPE.EQ.4)THEN
C
            WRITE(*,*)'  How many nodes do you want to add?'
            READ(*,*)NOP
            DO 330 I=1,NOP
```

```
               CALL GETNEWND(X1,Y1,Z1,I)
330          CONTINUE
             ITYPE=2
             NOP=NOP+2
             I1=1
C
             WRITE(*,2006)NE3
             READ(*,*)TENSN
             CALL GETSTR(NE3,ISTR3,BETA13,BETA23)
C
             WRITE(20,3001)ITYPE,NE3
             WRITE(20,3002)TENSN
             WRITE(20,3003)NOP
             WRITE(20,3004)X(NE1,1),Y(NE1,1),Z(NE1,1),I1
             DO 340 I=2,NOP-1
                WRITE(20,3004)X1(I-1),Y1(I-1),Z1(I-1),I
340          CONTINUE
             WRITE(20,3004)X(NE2,1),Y(NE2,1),Z(NE2,1),NOP
             WRITE(20,3005)ISTR3
             IF(ISTR3.NE.4)WRITE(20,3006)BETA13
             IF(ISTR3.EQ.4)WRITE(20,3007)BETA13,BETA23
C
           ENDIF
C
C
C      Get data for edge NE4:
C
405        WRITE(*,2002)NE4
C
           WRITE(*,*)'Enter the following:'
           WRITE(*,*)'  1  if grid points are specified'
           WRITE(*,*)'  2  if nodes for splining are specified'
           WRITE(*,*)'  3  if you want to let the end points of'
           WRITE(*,*)'     edges already entered define the curve'
           WRITE(*,*)'  4  if you want to use the end points of'
           WRITE(*,*)'     edges already defined but add (by '
           WRITE(*,*)'     typing in directly) some points in '
           WRITE(*,*)'     between'
           WRITE(*,*)
           WRITE(*,*)
           WRITE(*,*)
           WRITE(*,*)'Enter your choice'
           READ(*,*)ITYPE
C
           IF(ITYPE.NE.1 .AND. ITYPE.NE.2 .AND. ITYPE.NE.3
     /                   .AND. ITYPE.NE.4)GOTO 405
C
C
           IF(ITYPE.EQ.1)THEN
             CALL GETGRP(X1,Y1,Z1,NOP)
             IF(NOP.NE.JL)WRITE(*,*)
     /        'WARNING --- NUMBER OF GRID POINTS INCONSISTENT -',
     /        ' EDGE',NE4
             WRITE(20,3001)ITYPE,NE4
             DO 410 J=1,JL
                WRITE(20,3004)X1(J),Y1(J),Z1(J),J
410          CONTINUE
C
```

```
C
          ELSEIF(ITYPE.EQ.2)THEN
            CALL GETNODES(X1,Y1,Z1,NOP)
C
            WRITE(*,2006)NE4
            READ(*,*)TENSN
            CALL GETSTR(NE4,ISTR4,BETA14,BETA24)
C
            WRITE(20,3001)ITYPE,NE4
            WRITE(20,3002)TENSN
            WRITE(20,3003)NOP
            DO 420 I=1,NOP
               WRITE(20,3004)X1(I),Y1(I),Z1(I),I
420          CONTINUE
            WRITE(20,3005)ISTR4
            IF(ISTR4.NE.4)WRITE(20,3006)BETA14
            IF(ISTR4.EQ.4)WRITE(20,3007)BETA14,BETA24
C
          ELSEIF(ITYPE.EQ.3)THEN
            ITYPE=2
            NOP=2
            I1=1
            I2=2
C
            WRITE(*,2006)NE4
            READ(*,*)TENSN
            CALL GETSTR(NE4,ISTR4,BETA14,BETA24)
C
            WRITE(20,3001)ITYPE,NE4
            WRITE(20,3002)TENSN
            WRITE(20,3003)NOP
            IMAX=IL
            IF(TYPE(NE1).EQ.2)IMAX=NODES(NE1)
            WRITE(20,3004)X(NE1,IMAX),Y(NE1,IMAX),Z(NE1,IMAX),I1
            IMAX=IL
            IF(TYPE(NE2).EQ.2)IMAX=NODES(NE2)
            WRITE(20,3004)X(NE2,IMAX),Y(NE2,IMAX),Z(NE2,IMAX),I2
            WRITE(20,3005)ISTR4
            IF(ISTR4.NE.4)WRITE(20,3006)BETA14
            IF(ISTR4.EQ.4)WRITE(20,3007)BETA14,BETA24
C
C
          ELSEIF(ITYPE.EQ.4)THEN
C
            WRITE(*,*)'  How many nodes do you want to add?'
            READ(*,*)NOP
            DO 430 I=1,NOP
               CALL GETNEWND(X1,Y1,Z1,I)
430         CONTINUE
            ITYPE=2
            NOP=NOP+2
            I1=1
C
            WRITE(*,2006)NE4
            READ(*,*)TENSN
            CALL GETSTR(NE4,ISTR4,BETA14,BETA24)
C
            WRITE(20,3001)ITYPE,NE4
```

```
            WRITE(20,3002)TENSN
            WRITE(20,3003)NOP
            IMAX=IL
            IF(TYPE(NE1).EQ.2)IMAX=NODES(NE1)
            WRITE(20,3004)X(NE1,IMAX),Y(NE1,IMAX),Z(NE1,IMAX),I1
            DO 440 I=2,NOP-1
               WRITE(20,3004)X1(I-1),Y1(I-1),Z1(I-1),I
440         CONTINUE
            IMAX=IL
            IF(TYPE(NE2).EQ.2)IMAX=NODES(NE2)
            WRITE(20,3004)X(NE2,IMAX),Y(NE2,IMAX),Z(NE2,IMAX),I2
            WRITE(20,3005)ISTR4
            IF(ISTR4.NE.4)WRITE(20,3006)BETA14
            IF(ISTR4.EQ.4)WRITE(20,3007)BETA14,BETA24
C
          ENDIF
500     CONTINUE
      ENDIF
C
2001  FORMAT(/////,' Enter the TYPE for edge ',I2,
     $///,'          Enter:',//,
     $'         1   if grid points along the edge are to be ',
     $'specified',/,'          2   if nodes for splining are to be',
     $' specified',/)
2002  FORMAT(/////////////,'      SPECIFYING EDGE ',I2,///)
2003  FORMAT(/////,' Enter the TYPE for edge ',I2,
     $///,'          Enter:',//,
     $'         1   if grid points along the edge are to be ',
     $'specified',/,'          2   if nodes for splining are to be',
     $' specified',/,'          3   if end nodes of edges already ',
     $'entered   ',/,'              define the curve completely',/)
2006  FORMAT(///////,' Enter the TENSION parameter for curve ',I2)
C
3001  FORMAT(' ',I3,'     Type - EDGE NO: ',I2,
     $'-----------------------------------------')
3002  FORMAT(' ',F6.2,'    Tension parameter')
3003  FORMAT(' ',I3,'     Number of nodes')
3004  FORMAT(3X,3(F8.5,3X),'  --',I2)
3005  FORMAT(' ',I3,'     StretchType')
3006  FORMAT(' ',F8.4,'     Stretching parameter BETA')
3007  FORMAT(' ',2(F8.4,3X),'     Stretching parameters BETA1 and',
     $' BETA2')
3010  FORMAT(' ',I3,'     StretchType ',I2,' ------------------')
C
C
      STOP
      END
C
C===============================================================#
C
      SUBROUTINE GETNODES(X1,Y1,Z1,NOP)
C
C This subroutine reads in and arranges nodal points to define
C an edge.
C
      DIMENSION X1(100),Y1(100),Z1(100)
C
      INNUM0=10
```

```
      NOP=0
C
      WRITE(*,*)'How many sections is the edge composed of?'
      READ(*,*)NOSECT
C
      DO 100 ISECT=1,NOSECT
C
         INNUM=INNUM0+ISECT
         WRITE(*,200)ISECT,INNUM
         READ(INNUM,*)NP
         WRITE(*,201)NP
         READ(*,*)N1
         WRITE(*,202)
         READ(*,*)N2
C
         N12=IABS(N2-N1)+1
         NOP=NOP+N12
C
         IF(N2.GE.N1)THEN
            DO 10 J=1,N1-1
               READ(INNUM,*)
 10         CONTINUE
C
            I=NOP-N12
            DO 20 J=N1,N2
               I=I+1
               READ(INNUM,*)X1(I),Y1(I),Z1(I)
 20         CONTINUE
C
         ELSE
C
            DO 30 J=1,N2-1
               READ(INNUM,*)
 30         CONTINUE
C
            I=NOP+1
            DO 40 J=N2,N1
               I=I-1
               READ(INNUM,*)X1(I),Y1(I),Z1(I)
 40         CONTINUE
         ENDIF
C
         CLOSE(INNUM)
 100  CONTINUE
C
 200  FORMAT(' Section ',I2,' will be read in from UNIT',I3)
 201  FORMAT(' There are ',I2,' points in the file.',/
     / ,' Enter the number of the point that is to be the',/
     / ,' the first on the current section.')
 202  FORMAT(' Enter the number of the point that is to be the',/
     / ,' the last on the current section.')
C
      RETURN
      END
C
C===============================================================#
C
      SUBROUTINE GETGRP(X1,Y1,Z1,NOP)
```

```
C
C This subroutine reads in grid point coordinates for an
C edge and stores in either forward or reversed order.
C
      DIMENSION X1(100),Y1(100),Z1(100)
C
C
C
      INNUM0=10
      NOP=0
C
      WRITE(*,*)'How many sections is the edge composed of?'
      READ(*,*)NOSECT
C
      DO 500 ISECT=1,NOSECT
C
         INNUM=INNUM0+ISECT
         WRITE(*,200)ISECT,INNUM
         READ(INNUM,*)NP
         WRITE(*,201)NP
         READ(*,*)N1
         WRITE(*,202)
         READ(*,*)N2
C
         N12=IABS(N2-N1)+1
         NOP=NOP+N12
C
         IF(N2.GE.N1)THEN
            DO 10 J=1,N1-1
               READ(INNUM,*)
10          CONTINUE
C
            I=NOP-N12
            DO 20 J=N1,N2
               I=I+1
               READ(INNUM,*)X1(I),Y1(I),Z1(I)
20          CONTINUE
C
         ELSE
C
            DO 60 J=1,N2-1
               READ(INNUM,*)
60          CONTINUE
C
            I=NOP+1
            DO 70 J=N2,N1
               I=I-1
               READ(INNUM,*)X1(I),Y1(I),Z1(I)
70          CONTINUE
C
         ENDIF
         CLOSE(INNUM)
500   CONTINUE
C
200   FORMAT(' Section ',I2,' will be read in from UNIT',I3)
201   FORMAT(' There are ',I2,' grid points in the file.',/
     / ,' Enter the number of the grid point that is to be the',/
     / ,' the first or the current section.')
```

```
202   FORMAT(' Enter the number of the grid point that is to be the',/
     / ,' the last on the current section.')
      RETURN
      END
C
C
C==============================================================#
C
C
      SUBROUTINE GETNEWND(X1,Y1,Z1,I)
C
C This subroutine reads in from the screen new points that are
C to be included on the edge.
C
      DIMENSION X1(100),Y1(100),Z1(100)
C
      WRITE(*,2001)I
      READ(*,*)X1(I),Y1(I),Z1(I)
C
2001  FORMAT(' Please enter the x, y, and z coordinates for inter-'
     / ,/,' mediate point number ',I2)
      RETURN
      END
C
C
C===============================================================
C
      SUBROUTINE GETSTR(NE1,ISTR,BETA1,BETA2)
C
C This subroutine reads information regarding stretching function
C along edge NE1.
C
      WRITE(*,2011)NE1
      WRITE(*,2020)
100   READ(*,*)ISTR
      IF(ISTR.LT.4 .AND. ISTR.GT.0)THEN
         WRITE(*,2031)NE1
         READ(*,*)BETA1
      ELSEIF(ISTR.EQ.4)THEN
         WRITE(*,2036)NE1
         READ(*,*)BETA1,BETA2
      ELSEIF(ISTR.EQ.0)THEN
         BETA1=1.1
      ELSE
         WRITE(*,*)'   Please enter a number from 0 to 4'
         GOTO 100
      ENDIF
C
2011  FORMAT(' Enter the STRETCH TYPE for edge ',I2)
2020  FORMAT(/,' Enter:  0   for  no stretching',/,
     / '          1   for  concentration near lower boundary',/,
     / '          2   for  concentration near upper boundary',/,
     / '          3   for  concentration near both boundaries',/,
     / '                   (one parameter stretching function)',/,
     / '          4   for  concentration near both boundaries',/,
     / '                   (two-parameter stretching function)')
2031  FORMAT(' Enter the stretching parameter (BETA) for edge ',
     / I2)
2036  FORMAT(' Enter the stretching parameters (BETA1 and BETA2) ',
     / 'for edge ',I2)
C
C
      RETURN
      END
```

A.4 Listing of EDGE

```
      PROGRAM EDGE
C=====================================================================
C  This program generates grid points along an edge which is given    I
C  by a set of discrete nodes. The edge can be made up of up to       I
C  several sections. A parametric tension spline is fit through       I
C  each section.  Each section has its own control parameters         I
C  such as number of grid points, tension, and stretching functions. I
C  NOTE: Subroutines have been adopted from GRID3D (and modified      I
C        slightly) to generate the grid points on the curves.         I
C                                                                     I
C                                                                     I
C     PARAMETERS:                                                     I
C        MxBPts - Maximum number of nodes per section                 I
C        MxGSiz - Maximum number of grid points per section           I
C        MxSect - Maximum number of sections                          I
C        MxBCvs - Should not be modified                              I
C                                                                     I
C                                                                     I
C                                                                     I
C=====================================================================
        PARAMETER (MxBPts=31, MxGSiz=101, MxSect=5, MxBCvs=1)
C
      DIMENSION x(MxSect,MxBPts),  y(MxSect,MxBPts),
     $          z(MxSect,MxBPts),  zx(MxSect,MxBPts),
     $          zy(MxSect,MxBPts), zz(MxSect,MxBPts),
     $          s(MxSect,MxBPts), Tensn(MxSect)
C
      DIMENSION Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts)
C
      DIMENSION XB(MxGSiz,MxSect), YB(MxGSiz,MxSect),
     $          ZB(MxGSiz,MxSect), StrB(MxGSiz,MxBCvs)
C
      INTEGER NDPts(MxSect), ILS(MxSect), StrTp
C
C
      WRITE(*,*)'                           Running PROGRAM EDGE'
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,*)' The input data will be read in from UNIT 1'
      WRITE(*,*)' The output will be written into UNIT 20'
      WRITE(*,*)
      WRITE(*,*)
      READ(1,*)NOSECT
C
C
        IL=0
        DO 200 is=1,NOSECT
           CALL RdSctIn(x,y,z,NDPts,is,Tensn,1,MxBPts,MxSect,
     $                StrTp,Beta1,Beta2,ILS(is))
           CALL PTSpln(x,y,z,s,zx,zy,zz,Diag,OfDiag,Right,NDPts,
     $                Tensn(is),is,MxBPts,MxSect)
           CALL CalcStr2(1,ILS(is),StrTp,Beta1,Beta2,
     $                  StrB,MxBCvs,MxGSiz)
```

```
            CALL EdgGPts(is,1,ILS(is),XB,YB,ZB,StrB,x,y,z,s,
     $                   zx,zy,zz,NDPts,Tensn(is),
     $                   MxBCvs,MxBPts,MxGSiz,MxSect)
             IL=IL+ILS(is)
200       CONTINUE
          IL=IL-(NOSECT-1)
          WRITE(20,*)IL,'    Number of grid points'
          IP=0
          DO 301 is=1,NOSECT
             DO 300 i=1,ILS(is)-1
                IP=IP+1
                WRITE(20,3001)XB(i,is),YB(i,is),ZB(i,is),IP
300          CONTINUE
301       CONTINUE
          is=NOSECT
          i=ILS(NOSECT)
          WRITE(20,3001)XB(i,is),YB(i,is),ZB(i,is),IL
C
C
3001  FORMAT(' ',3X,3(F9.6,3X),'  --- ',I3)
      STOP
      END
C
C==================================================================C
C
      SUBROUTINE RdSctIn (x,y,z,NDPts,CrvNum,Tensn,InNum,MxBPts,MxSect,
     $                   StrTp,Beta1,Beta2,IL)
C
C This SUBROUTINE reads in the information concerning discrete points on
C the boundaries.  This information is used for generating spline-fitted
C boundary approximation curves.
C
      INTEGER  CrvNum, i, NDPts(MxSect), InNum, StrTp, IL
C
      REAL  x(MxSect,MxBPts), y(MxSect,MxBPts),
     $      z(MxSect,MxBPts), Tensn(MxSect)
C
      READ(InNum,*) IL
      READ(InNum,*) Tensn(CrvNum)
      READ(InNum,*) NDPts(CrvNum)
C
      DO 10 i=1,NDPts(CrvNum)
         READ(InNum,*) x(CrvNum,i), y(CrvNum,i), z(CrvNum,i)
 10   CONTINUE
C
      READ(InNum,*)StrTp
      IF(StrTp.NE.4)THEN
         READ(InNum,*)Beta1
      ELSE
         READ(InNum,*)Beta1,Beta2
      ENDIF
C
      RETURN
      END
C
C==================================================================C
      SUBROUTINE CalcS (x,y,z,s,NDPts,CrvNum,MxBPts,MxSect)
C
```

```
C This SUBROUTINE calculates the spline parameter, s, as an approximate
C arc length.
C
      INTEGER  NDPts(MxSect), CrvNum, i
C
      REAL  x(MxSect,MxBPts), y(MxSect,MxBPts),
     $      z(MxSect,MxBPts), s(MxSect,MxBPts)
C
      s(CrvNum,1)=0.0
C
      DO 10 i=2,NDPts(CrvNum)
       s(CrvNum,i)=s(CrvNum,i-1)
     $                  +SQRT( (x(CrvNum,i)-x(CrvNum,i-1))**2
     $                        +(y(CrvNum,i)-y(CrvNum,i-1))**2
     $                        +(z(CrvNum,i)-z(CrvNum,i-1))**2)
 10   CONTINUE
C
      RETURN
      END
C
C==================================================================C
      SUBROUTINE SplMat (Diag,OfDiag,Right,w,s,NDPts,T,CrvNum,
     $                   MxBPts,MxSect)
C
C This SUBROUTINE forms the parametric tension spline matrix for a
C particular boundary curve data set.
C
      INTEGER  i, NDPts(MxSect), CrvNum
C
      REAL Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts),
     $     w(MxSect,MxBPts), s(MxSect,MxBPts), T, h, hm
C
      Diag(1)=1.0
      OfDiag(1)=0.0
      Right(1)=0.0
C
      DO 10 i=2,NDPts(CrvNum)-1
         h=s(CrvNum,i+1)-s(CrvNum,i)
         hm=s(CrvNum,i)-s(CrvNum,i-1)
         Diag(i)=(T*COSH(T*hm)/SINH(T*hm)-1/hm+T*COSH(T*h)/SINH(T*h)
     $           -1/h)/T**2
         OfDiag(i)=(1/h-T/SINH(T*h))/T**2
         Right(i)= (w(CrvNum,i+1)-w(CrvNum,i))/h
     $            -(w(CrvNum,i)-w(CrvNum,i-1))/hm
 10   CONTINUE
C
      Diag(NDPts(CrvNum))=1.0
      OfDiag(NDPts(CrvNum)-1)=0.0
      Right(NDPts(CrvNum))=0.0
C
      RETURN
      END
C
C==================================================================C
      SUBROUTINE SplSlv (Diag,OfDiag,Right,Derv2,NDPts,CrvNum,
     $                   MxBPts,MxSect)
C
C This SUBROUTINE solves the diagonally dominant parametric tension
```

```
C spline matrix for a given data set using the Gauss-Seidel iteration.
C Convergence is assumed after 20 iterations.
C
      INTEGER  i, j, NDPts(MxSect), CrvNum
C
      REAL Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts),
     $     Derv2(MxSect,MxBPts)
C
C Initialize the second derivative matrix to all zeroes.
C
      DO 10 i=1,NDPts(CrvNum)
         Derv2(CrvNum,i)=0.0
 10   CONTINUE
C
C Calculate the second derivative values using 20 iterations of
C the Gauss-Seidel method.
C
      DO 30 j=1,20
         DO 20 i=2,NDPts(CrvNum)-1
           Derv2(CrvNum,i)=(Right(i)-OfDiag(i)*Derv2(CrvNum,i+1)
     $                          -OfDiag(i-1)*Derv2(CrvNum,i-1))
     $                          /Diag(i)
 20      CONTINUE
 30   CONTINUE
C
      RETURN
      END
C
C===================================================================C
      FUNCTION SplVal (s,w,Derv2,sval,T,n,CrvNum,MxBPts,MxSect)
C
C This real function finds the w-value (x-value or y-value) corresponding
C to a specified s-value using the parametric tension spline curve
C generated for a particular boundary curve data set.
C
      INTEGER  n, CrvNum
C
      REAL  s(MxSect,MxBPts), w(MxSect,MxBPts), Derv2(MxSect,MxBPts),
     $      sval, T, h, Interim, Temp1, Temp2
C
C
      Temp1=sval-s(CrvNum,n)
      h=s(CrvNum,n+1)-s(CrvNum,n)
      Temp2=s(CrvNum,n+1)-sval
      Interim=Derv2(CrvNum,n)/T**2*SINH(T*Temp2)/SINH(T*h)
     $        +(w(CrvNum,n)-Derv2(CrvNum,n)/T**2)*Temp2/h
      SplVal=Interim+Derv2(CrvNum,n+1)/T**2*SINH(T*Temp1)
     $                    /SINH(T*h)+(w(CrvNum,n+1)
     $                    -Derv2(CrvNum,n+1)/T**2)*Temp1/h
C
      RETURN
      END
C
C===================================================================C
      SUBROUTINE PTSpln(x,y,z,s,XDerv2,YDerv2,ZDerv2,Diag,OfDiag,
     $                  Right,NDPts,Tensn,CrvNum,MxBPts,MxSect)
C
C This SUBROUTINE forms the main routine for the parametric tension
```

```
C spline process.
C
       INTEGER  NDPts(MxSect), CrvNum
C
       REAL Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts),
     $      XDerv2(MxSect,MxBPts), YDerv2(MxSect,MxBPts),
     $      ZDerv2(MxSect,MxBPts), Tensn,
     $      x(MxSect,MxBPts), y(MxSect,MxBPts),
     $      z(MxSect,MxBPts), s(MxSect,MxBPts)
C
C
       CALL CalcS(x,y,z,s,NDPts,CrvNum,MxBPts,MxSect)
       CALL SplMat(Diag,OfDiag,Right,x,s,NDPts,Tensn,CrvNum,
     $ -            MxBPts,MxSect)
       CALL SplSlv(Diag,OfDiag,Right,XDerv2,NDPts,CrvNum,MxBPts,MxSect)
       CALL SplMat(Diag,OfDiag,Right,y,s,NDPts,Tensn,CrvNum,
     $              MxBPts,MxSect)
       CALL SplSlv(Diag,OfDiag,Right,YDerv2,NDPts,CrvNum,MxBPts,MxSect)
       CALL SplMat(Diag,OfDiag,Right,z,s,NDPts,Tensn,CrvNum,
     $              MxPts,MxSect)
       CALL SplSlv(Diag,OfDiag,Right,ZDerv2,NDPts,CrvNum,MxBPts,MxSect)
C
       RETURN
       END
C
C==================================================================C
       SUBROUTINE SplInt(n,s,SValue,NDPts,CurCrv,MxBPts,MxSect)
C
C This SUBROUTINE finds the proper interval in which a point on a specified
C boundary lies.  The interval indicates which initial data points the
C point in question lies between and thus which spline coefficients to
C use.
C
C
       INTEGER  i, n, CurCrv, NDPts(MxSect)
C
       REAL  Temp, SValue, s(MxSect,MxBPts)
C
       n=1
       i=NDPts(CurCrv)
C
 10    IF ((n.EQ.1).AND.(i.GT.1)) THEN
          I=I-1
          Temp=SValue-s(CurCrv,i)
C
          IF (Temp.GT.0.0) THEN
             n=i
          ENDIF
C
          GOTO 10
       ENDIF
C
       RETURN
       END
C
C==================================================================C
       SUBROUTINE FAlNew(AlNew,Alpha,B,Str)
C
```

```
C This SUBROUTINE computes the new Alpha value after stretching as
C AlNew.  Alpha is a dummy variable representing either Xi, Eta or Zeta.
C
      INTEGER  Str
C
      REAL  Alpha, Temp1, Temp2, B2, AlNew, B
C
      AlNew=Alpha
      Temp1=(B+1)/(B-1)
C
      IF (Str.EQ.1) THEN
         Temp2=Temp1**(1-Alpha)
         AlNew=((B+1)-(B-1)*Temp2)/(Temp2+1)*1
      ENDIF
C
      IF (Str.EQ.2) THEN
         B2=0
         Temp2=Temp1**((Alpha-B2)/(1-B2))
         AlNew=((B+2*B2)*Temp2-B+2*B2)/((2*B2+1)*(1+Temp2))
      ENDIF
C
      IF (Str.EQ.3) THEN
         B2=0.5
         Temp2=Temp1**((Alpha-B2)/(1-B2))
         AlNew=((B+2*B2)*Temp2-B+2*B2)/((2*B2+1)*(1+Temp2))
      ENDIF
C
      RETURN
      END
C
C===================================================================C
      SUBROUTINE CalcStr2(EdgNum,NGPts,StrTp,Beta1,Beta2,
     $                    StrB,MxBCvs,MxGSiz)
C
C This subroutine calculates the distribution function base on the
C stretching parameters 'StrTp' and 'Beta'
C
      INTEGER NGPts, StrTp, EdgNum, i
C
      REAL    StrB(MxGSiz,MxBCvs), Beta1, Beta2, A, B, DZ
C
      StrB(1,EdgNum)=0.
      IF(StrTp.LE.3)THEN
         DO 10 i=1,NGPts-1
            Alpha=(i-1.)/(NGPts-1.)
            CALL FAlNew(AlNew,Alpha,Beta1,StrTp)
            StrB(i,EdgNum)=AlNew
10       CONTINUE
      ELSEIF(StrTp.EQ.4)THEN
         CALL Str4Prm(Beta1,Beta2,A,B,DZ)
         DO 20 i=2,NGPts-1
            Alpha=(i-1.)/(NGPts-1.)
            CALL Str4(AlNew,Alpha,A,B,DZ)
            StrB(i,EdgNum)=AlNew
20       CONTINUE
      ENDIF
      StrB(NGPts,EdgNum)=1.
C
```

```
      RETURN
      END
C
C=====================================================================C
      SUBROUTINE EdgGPts(CrvNum,EdgNum,NGPts,XB,YB,ZB,StrB,
     $                   x,y,z,s,zx,zy,zz,NDPts,Tensn,
     $                   MxBCvs,MxBPts,MxGSiz,MxSect)
C
C This subroutine calculates the grid point location along an edge
C based on a spline curve fitted through specified nodal points and a
C given distribution function.
C
      INTEGER CrvNum, EdgNum, NGPts, NDPts(MxSect), i, n
C
      REAL    XB(MxGSiz,MxSect), YB(MxGSiz,MxSect), ZB(MxGSiz,MxSect),
     $        StrB(MxGSiz,MxBCvs), x(MxSect,MxBPts), y(MxSect,MxBPts),
     $        z(MxSect,MxBPts), zx(MxSect,MxBPts), zy(MxSect,MxBPts),
     $        zz(MxSect,MxBPts), s(MxSect,MxBPts), Tensn
C
      SRa=S(CrvNum,NDPts(CrvNum))
C
      DO 10 i=1,NGPts
         SB=SRa*StrB(i,EdgNum)
         CALL SplInt(n,s,SB,NDPts,CrvNum,MxBPts,MxSect)
         XB(i,CrvNum)=SplVal(s,x,zx,SB,Tensn,n,CrvNum,MxBPts,MxSect)
         YB(i,CrvNum)=SplVal(s,y,zy,SB,Tensn,n,CrvNum,MxBPts,MxSect)
         ZB(i,CrvNum)=SplVal(s,z,zz,SB,Tensn,n,CrvNum,MxBPts,MxSect)
10    CONTINUE
C
      RETURN
      END
C
C=====================================================================C
      SUBROUTINE Str4Prm(S0,S1,A,B,DZ)
C
      REAL S0, S1, A, B, DZ, Y, PI
C
C  This subroutine calculates the parameters A, B, and DZ for the two-
C  sided Vinokur stretching function.
C
      PI=ACOS(-1.)
C
      A=SQRT(S0/S1)
      B=SQRT(S0*S1)
C
      IF(B.GT.1.001)THEN
         IF(B.LE.2.7829681)THEN
            Y=B-1
            DZ=SQRT(6.*Y)*(1.-0.15*Y+0.057321429*(Y**2)
     $                    -0.024907295*(Y**3)+0.0077424461*(Y**4)
     $                    -0.0010794123*(Y**5))
         ELSEIF(B.GT.2.7829681)THEN
            V=LOG(B)
            W=1./B - 0.028527431
            DZ=V+(1.+1./V)*LOG(2.*V)-0.02041793+0.24902722*W
     $         +1.9496443*(W**2)-2.6294547*(W**3)+8.56795911*(W**4)
         ENDIF
      ELSEIF(B.LT.0.999)THEN
```

```
         IF(B.LE.0.26938972)THEN
            DZ=PI*(1.-B+B**2-(1.+(PI**2)/6.)*(B**3)+6.794732*(B**4)
     $         -13.205501*(B**5)+11.726095*(B**6))
         ELSE
            Y=B-1
            DZ=SQRT(6.*Y)*(1.+0.15*Y+0.057321429*(Y**2)
     $                        +0.048774238*(Y**3)-0.053337753*(Y**4)
     $                        +0.075845134*(Y**5))
         ENDIF
      ENDIF
C
C
      RETURN
      END
C
C=====================================================================C
      SUBROUTINE Str4(AlNew,Alpha,A,B,DZ)
C
      REAL AlNew, Alpha, A, B, DZ, U, T
C
C  This subroutine calculates the value of the two-sided Vinokur
C  stretching function based on the value of the parameters A, B,
C  and DZ, and on the value of the "computational" coordinate Alpha.
C
C
      IF(B.GT.1.001)THEN
        U=0.5+TANH(DZ*(Alpha-0.5))/(2.*TANH(DZ/2.))
      ELSEIF(B.LT.0.999)THEN
        U=0.5+TAN(DZ*(Alpha-0.5))/(2.*TAN(DZ/2.))
      ELSE
        U=Alpha*(1.+2.*(B-1)*(Alpha-0.5)*(1-Alpha))
      ENDIF
      T=U/(A+(1.-A)*U)
      AlNew=T
C
C
      RETURN
      END
```

A.5 Listing of EDGPREP

```
      PROGRAM EDGPREP
C
       PARAMETER (MxNode=100)
C
       INTEGER ISTR,ILS,NOP
C
       REAL  TENSION,BETA1,BETA2,
      /      X1(MxNode),Y1(MxNode),Z1(MxNode)
C
C This program prepares input files for EDGE by reading the
C necessary information from the screen and from files.
C
       WRITE(*,*)
       WRITE(*,*)
       WRITE(*,*)
       WRITE(*,*)'This program prepares an input file for the program'
       WRITE(*,*)'EDGE.  EDGE generates the grid point coordinates on'
       WRITE(*,*)'a curve given by a set of nodes through which'
       WRITE(*,*)'a spline curve can be fitted.  The spline curve can'
       WRITE(*,*)'be made up from several sections.'
       WRITE(*,*)
       WRITE(*,*)'Enter the number of sections'
       WRITE(*,*)
       READ(*,*)NOSECT
C
       WRITE(*,*)
       WRITE(*,*)
       WRITE(*,*)'The data will be written into UNIT 20'
       WRITE(20,*)NOSECT,'       Number of sections'
C
C
       DO 100 is=1,NOSECT
         WRITE(*,*)
         WRITE(*,*)
         WRITE(*,*)
         WRITE(*,*)
         WRITE(*,*)'Now enter data for section number',is
         WRITE(*,*)
         WRITE(*,*)'Enter number of grid points to be on the section'
         WRITE(*,*)
         READ(*,*)ILS
         CALL GETNODES(X1,Y1,Z1,NOP,MxNode)
C
         WRITE(*,2001)is
         READ(*,*)TENSION
         CALL GETSTR(is,ISTR,BETA1,BETA2)
C
         WRITE(20,3001)ILS,is
         WRITE(20,3002)TENSION
         WRITE(20,3003)NOP
         DO 20 I=1,NOP
            WRITE(20,3004)X1(I),Y1(I),Z1(I),I
 20      CONTINUE
         WRITE(20,3005)ISTR
```

```
        IF(ISTR.NE.4)WRITE(20,3006)BETA1
        IF(ISTR.EQ.4)WRITE(20,3007)BETA1,BETA2
C
100   CONTINUE
C
C
C
2001  FORMAT(//,' Enter the TENSION parameter for section ',I2)
3001  FORMAT(' ',I3,'   No of gridpts: Section ',I2,
     $'---------------')
3002  FORMAT(' ',F6.2,'    Tension parameter')
3003  FORMAT(' ',I3,'     Number of nodes')
3004  FORMAT(3X,3(F8.5,3X),'  ---',I2)
3005  FORMAT(' ',I3,'     StretchType')
3006  FORMAT(' ',F8.4,'     Stretching parameter BETA')
3007  FORMAT(' ',2(F8.4,3X),'     Stretching parameters BETA1 and',
     $' BETA2')
C
C
      STOP
      END
C
C================================================================#
C
      SUBROUTINE GETNODES(X1,Y1,Z1,NOP,MxNode)
C
C This subroutine reads in and arranges nodal points to define
C two parallel curve sections (one on each blade surface).
C
      DIMENSION X1(MxNode),Y1(MxNode),Z1(MxNode)
C
      INNUM0=10
      NOP=0
      PI=ACOS(-1.)
C
      WRITE(*,*)'The data for the section can be read in several'
      WRITE(*,*)'parts, where each part is a single node point or a'
      WRITE(*,*)'series of node points.  Note, the node points can'
      WRITE(*,*)'be read in both forward and reverse order from'
      WRITE(*,*)'the input files.'
      WRITE(*,*)
      WRITE(*,*)'How many parts is the section composed of?'
      READ(*,*)NOSECT
C
      DO 100 ISECT=1,NOSECT
C
         INNUM=INNUM0+ISECT
         WRITE(*,200)ISECT,INNUM
         READ(INNUM,*)NP
         WRITE(*,201)NP
         READ(*,*)N1
         WRITE(*,202)
         READ(*,*)N2
C
         N12=IABS(N2-N1)+1
         NOP=NOP+N12
C
         IF(N2.GE.N1)THEN
```

```
            DO 10 J=1,N1-1
               READ(INNUM,*)
10          CONTINUE
C
            I=NOP-N12
            DO 20 J=N1,N2
               I=I+1
               READ(INNUM,*)X1(I),Y1(I),Z1(I)
20          CONTINUE
C
         ELSE
C
            DO 30 J=1,N2-1
               READ(INNUM,*)
30          CONTINUE
C
            I=NOP+1
            DO 40 J=N2,N1
               I=I-1
               READ(INNUM,*)X1(I),Y1(I),Z1(I)
40          CONTINUE
         ENDIF
C
         CLOSE(INNUM)
100   CONTINUE
C
200   FORMAT(' Part ',I2,' will be read in from UNIT',I3)
201   FORMAT(' There are ',I2,' node points in the file.',/
     / ,' Enter the number of the node point that is to be the',/
     / ,' the first on the current part.')
202   FORMAT(' Enter the number of the node point that is to be the',/
     / ,' the last on the current part.')
C
      RETURN
      END
C
C==============================================================
C
      SUBROUTINE GETSTR(NE1,ISTR,BETA1,BETA2)
C
C This subroutine reads information regarding the stretching function
C along curve section NE1.
C
      WRITE(*,2011)NE1
      WRITE(*,2020)
100   READ(*,*)ISTR
      IF(ISTR.LT.4 .AND. ISTR.GT.0)THEN
         WRITE(*,2031)NE1
         READ(*,*)BETA1
      ELSEIF(ISTR.EQ.4)THEN
         WRITE(*,2036)NE1
         READ(*,*)BETA1,BETA2
      ELSEIF(ISTR.EQ.0)THEN
         BETA1=1.1
      ELSE
         WRITE(*,*)'   Please enter a number from 0 to 4'
         GOTO 100
      ENDIF
```

```
C
2011  FORMAT(' Enter the STRETCH TYPE for section',I2)
2020  FORMAT(/,' Enter:  0   for  no stretching',/,
     / '          1   for  concentration near lower boundary',/,
     / '          2   for  concentration near upper boundary',/,
     / '          3   for  concentration near both boundaries',/,
     / '          4   for  two-parameter stretching function')
2031  FORMAT(' Enter the stretching parameter (BETA) for section ',
     / I2)
2036  FORMAT(' Enter the stretching parameters (BETA1 and BETA2) ',
     / 'for section ',I2)
C
      RETURN
      END
```

A.6 Listing of GRIDTST

```
      PROGRAM GRIDTST
      PARAMETER (IM=11,JM=51,KM=151)
C
C=====================================================================
C   This program is used to test whether all Jacobians for a grid    I
C   system are positive.                                             I
C=====================================================================
C
C
C
      DIMENSION X(IM,JM,KM),Y(IM,JM,KM),Z(IM,JM,KM)
C
      DIMENSION DXDXI(IM),DXDET(IM),DXDZET(IM),DYDXI(IM),DYDET(IM)
     / ,DYDZET(IM),DZDXI(IM),DZDET(IM),DZDZET(IM)
C
C
C    Read in the grid point coordinates.
      READ(1,*)IL
      READ(1,*)JL
      READ(1,*)KL
      DO 5 I=1,IL
      DO 5 J=1,JL
      DO 5 K=1,KL
         READ(1,*)X(I,J,K),Y(I,J,K),Z(I,J,K)
5     CONTINUE
C
      DXI=1./FLOAT(IL)
      DET=1./FLOAT(JL)
      DZET=1./FLOAT(KL)
C
C
C     Calculate the metric coefficents at the regular grid points.
C
      INEG=0
      DO 30 K=1,KL
      DO 20 J=1,JL
         CALL DDXIJK(DXI,IL,JL,KL,X,Y,Z,J,K
     /               ,DXDXI,DYDXI,DZDXI,IM,JM,KM)
         CALL DDETJK(DET,IL,JL,KL,X,Y,Z,J,K
     /               ,DXDET,DYDET,DZDET,IM,JM,KM)
         CALL DDZETJK(DZET,IL,JL,KL,X,Y,Z,J,K
     /               ,DXDZET,DYDZET,DZDZET,IM,JM,KM)
C
         DO 10 I=1,IL
            RJACB=
     /      (DXDXI(I)*(DYDET(I)*DZDZET(I)-DZDET(I)*DYDZET(I))
     /      +DYDXI(I)*(DZDET(I)*DXDZET(I)-DXDET(I)*DZDZET(I))
     /      +DZDXI(I)*(DXDET(I)*DYDZET(I)-DYDET(I)*DXDZET(I)))
            IF(RJACB.LE.0)WRITE(2,50)I,J,K,RJACB
            IF(RJACB.LE.0)INEG=INEG+1
10       CONTINUE
20    CONTINUE
30    CONTINUE
      IF(INEG.GT.0)WRITE(*,*)'      NEGATIVE JACOBIANS FOUND'
```

```
C
C
      RETURN
50    FORMAT(' ','(i,j,k)=(',2(i3,','),i3,')','    J=',e16.8)
      END
C
C
C
      SUBROUTINE DDXIJK(DXI,IL,JL,KL,X,Y,Z,J,K,DXDXI
     / ,DYDXI,DZDXI,IM,JM,KM)
C
C=====================================================================
C      This subroutine calculates the derivatives of X, Y, and Z      I
C      (i.e., the coordinates of the Cartesian coordinate system)     I
C      with respect to the coordinate XI of the transformed           I
C      coordinates.                                                   I
C=====================================================================
C
      DIMENSION X(IM,JM,KM),Y(IM,JM,KM),Z(IM,JM,KM)
     / ,DXDXI(IM),DYDXI(IM),DZDXI(IM)
C
C
C
      DXI2=2.*DXI
      DXDXI(1)=(-X(3,J,K)+4.*X(2,J,K)-3.*X(1,J,K))/DXI2
      DYDXI(1)=(-Y(3,J,K)+4.*Y(2,J,K)-3.*Y(1,J,K))/DXI2
      DZDXI(1)=(-Z(3,J,K)+4.*Z(2,J,K)-3.*Z(1,J,K))/DXI2
C
      DO 10 I=2,IL-1
         DXDXI(I)=(X(I+1,J,K)-X(I-1,J,K))/DXI2
         DYDXI(I)=(Y(I+1,J,K)-Y(I-1,J,K))/DXI2
         DZDXI(I)=(Z(I+1,J,K)-Z(I-1,J,K))/DXI2
10    CONTINUE
C
      DXDXI(IL)=(3.*X(IL,J,K)-4.*X(IL-1,J,K)
     /                      + X(IL-2,J,K))/DXI2
      DYDXI(IL)=(3.*Y(IL,J,K)-4.*Y(IL-1,J,K)
     /                      + Y(IL-2,J,K))/DXI2
      DZDXI(IL)=(3.*Z(IL,J,K)-4.*Z(IL-1,J,K)
     /                      + Z(IL-2,J,K))/DXI2
C
C
      RETURN
      END
C
C
      SUBROUTINE DDETJK(DET,IL,JL,KL,X,Y,Z,J,K,DXDET
     / ,DYDET,DZDET,IM,JM,KM)
C
C=====================================================================
C      This subroutine calculates the derivatives of X, Y, and Z      I
C      (i.e., the coordinates of the Cartesian coordinate system)     I
C      with respect to the coordinate ETA of the transformed          I
C      coordinates.                                                   I
C=====================================================================
C
      DIMENSION X(IM,JM,KM),Y(IM,JM,KM),Z(IM,JM,KM)
     / ,DXDET(IM),DYDET(IM),DZDET(IM)
```

```
C
C
      DET2=2.*DET
      IF(J.EQ.1)THEN
         DO 10 I=1,IL
            DXDET(I)=(-X(I,J+2,K)+4.*X(I,J+1,K)
     /                          -3.*X(I,J,K))/DET2
            DYDET(I)=(-Y(I,J+2,K)+4.*Y(I,J+1,K)
     /                          -3.*Y(I,J,K))/DET2
            DZDET(I)=(-Z(I,J+2,K)+4.*Z(I,J+1,K)
     /                          -3.*Z(I,J,K))/DET2
10       CONTINUE
C
      ELSE IF(J.EQ.JL)THEN
         DO 20 I=1,IL
            DXDET(I)=(3.*X(I,J,K)  - 4.*X(I,J-1,K)
     /                              +X(I,J-2,K))/DET2
            DYDET(I)=(3.*Y(I,J,K)  - 4.*Y(I,J-1,K)
     /                              +Y(I,J-2,K))/DET2
            DZDET(I)=(3.*Z(I,J,K)  - 4.*Z(I,J-1,K)
     /                              +Z(I,J-2,K))/DET2
20       CONTINUE
C
      ELSE
         DO 30 I=1,IL
            DXDET(I)=(X(I,J+1,K)-X(I,J-1,K))/DET2
            DYDET(I)=(Y(I,J+1,K)-Y(I,J-1,K))/DET2
            DZDET(I)=(Z(I,J+1,K)-Z(I,J-1,K))/DET2
30       CONTINUE
C
      ENDIF
C
C
      RETURN
      END
C
C
      SUBROUTINE DDZETJK(DZET,IL,JL,KL,X,Y,Z,J,K,DXDZET
     / ,DYDZET,DZDZET,IM,JM,KM)
C
C======================================================================
C      This subroutine calculates the derivatives of X, Y, and Z       I
C      (i.e., the coordinates of the Cartesian coordinate system)      I
C      with respect to the coordinate ZETA of the transformed          I
C      coordinates.                                                    I
C======================================================================
C
      DIMENSION X(IM,JM,KM),Y(IM,JM,KM),Z(IM,JM,KM)
     / ,DXDZET(IM),DYDZET(IM),DZDZET(IM)
C
C
      DZET2=2.*DZET
      IF(K.EQ.1)THEN
         DO 10 I=1,IL
            DXDZET(I)=(-X(I,J,K+2)+4.*X(I,J,K+1)
     /                          -3.*X(I,J,K))/DZET2
            DYDZET(I)=(-Y(I,J,K+2)+4.*Y(I,J,K+1)
     /                          -3.*Y(I,J,K))/DZET2
```

```
          DZDZET(I)=(-Z(I,J,K+2)+4.*Z(I,J,K+1)
     /                           -3.*Z(I,J,K))/DZET2
10      CONTINUE
C
      ELSE IF(K.EQ.KL)THEN
        DO 20 I=1,IL
          DXDZET(I)=(3.*X(I,J,K)  - 4.*X(I,J,K-1)
     /                            +X(I,J,K-2))/DZET2
          DYDZET(I)=(3.*Y(I,J,K)  - 4.*Y(I,J,K-1)
     /                            +Y(I,J,K-2))/DZET2
          DZDZET(I)=(3.*Z(I,J,K)  - 4.*Z(I,J,K-1)
     /                            +Z(I,J,K-2))/DZET2
20      CONTINUE
C
      ELSE
        DO 30 I=1,IL
          DXDZET(I)=(X(I,J,K+1)-X(I,J,K-1))/DZET2
          DYDZET(I)=(Y(I,J,K+1)-Y(I,J,K-1))/DZET2
          DZDZET(I)=(Z(I,J,K+1)-Z(I,J,K-1))/DZET2
30      CONTINUE
C
      ENDIF
C
C
      RETURN
      END
```

Appendix B -- LISTING OF PROGRAM GRID3D-v2

```
      PROGRAM GRID3D
C
C
       PARAMETER (MxSrfs=4, MxBCvs=16, MxBPts=21, MxGSiz=31)
C
C This SUBROUTINE generates a three-dimensional grid system using the
C "two-boundary" or "four-boundary" algebraic grid generation techniques.
C Boundary surface edge curves are formed from sets of nodal points by
C using parametric tension splines.  Boundary surfaces are formed by
C using the "bi-directional 3-D Hermite interpolation" technique.
C
       INTEGER  CrvNum, SrfNum, NSurfs, InNum, OutNum,
      $         StrXi, StrEt, StrZt, StrAA, StrBB, II, JJ, KK, AL, BL,
      $         i, j, k, NDPts(4), AAL(MxSrfs), BBL(MxSrfs),
      $         NGPts(MxBCvs), Type, StrTp, EdgNum, ZoneNo
C
       REAL  EtStep, XiStep, ZtStep, AAStep, BBStep,
      $      SigmaXi, SigmaEt, SigmaZt,
      $      KXi1, KXi2, KEta1, KEta2, KZeta1, KZeta2,
      $      BetaXi, BetaEt, BetaZt, BetaAA, BetaBB,
      $      h1(MxGSiz), h2(MxGSiz), h3(MxGSiz), h4(MxGSiz),
      $      h5(MxGSiz), h6(MxGSiz), h7(MxGSiz), h8(MxGSiz),
      $      kS(MxSrfs,MxGSiz,MxGSiz), k1(MxSrfs,MxGSiz),
      $      k2(MxSrfs,MxGSiz), k3(MxSrfs,MxGSiz), k4(MxSrfs,MxGSiz),
      $      SigmaAA(MxSrfs), SigmaBB(MxSrfs),
      $      XB(MxGSiz,4),YB(MxGSiz,4),ZB(MxGSiz,4),
      $      X1(MxGSiz), X2(MxGSiz), X3(MxGSiz), X4(MxGSiz),
      $      Y1(MxGSiz), Y2(MxGSiz), Y3(MxGSiz), Y4(MxGSiz),
      $      Z1(MxGSiz), Z2(MxGSiz), Z3(MxGSiz), Z4(MxGSiz),
      $      StrB(MxGSiz,MxBCvs),
      $      EtSt11(MxGSiz),EtSt12(MxGSiz),EtSt15(MxGSiz),
      $      EtSt16(MxGSiz),ZtSt1(MxGSiz),ZtSt2(MxGSiz),
      $      ZtSt5(MxGSiz),ZtSt6(MxGSiz)
C
       REAL
      $      PXS1PE(MxGSiz,MxGSiz), PXS2PE(MxGSiz,MxGSiz),
      $      PYS1PE(MxGSiz,MxGSiz), PYS2PE(MxGSiz,MxGSiz),
      $      PZS1PE(MxGSiz,MxGSiz), PZS2PE(MxGSiz,MxGSiz),
      $      PXS3Zt(MxGSiz,MxGSiz), PXS4Zt(MxGSiz,MxGSiz),
      $      PYS3Zt(MxGSiz,MxGSiz), PYS4Zt(MxGSiz,MxGSiz),
      $      PZS3Zt(MxGSiz,MxGSiz), PZS4Zt(MxGSiz,MxGSiz)
       REAL  Tensn(4),
      $      Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts),
      $      XDerv2(4,MxBPts), YDerv2(4,MxBPts),
      $      ZDerv2(4,MxBPts),
      $      x(4,MxBPts),  y(4,MxBPts),
      $      z(4,MxBPts),  s(4,MxBPts),
      $     zx(4,MxBPts), zy(4,MxBPts),
      $     zz(4,MxBPts)
       REAL  PX1PBB(MxGSiz), PX2PBB(MxGSiz),
      $      PY1PBB(MxGSiz), PY2PBB(MxGSiz),
      $      PZ1PBB(MxGSiz), PZ2PBB(MxGSiz),
      $      PX1PAA(MxGSiz), PX2PAA(MxGSiz),
      $      PY1PAA(MxGSiz), PY2PAA(MxGSiz),
      $      PZ1PAA(MxGSiz), PZ2PAA(MxGSiz),
```

```
     $       PX3PBB(MxGSiz), PX4PBB(MxGSiz),
     $       PY3PBB(MxGSiz), PY4PBB(MxGSiz),
     $       PZ3PBB(MxGSiz), PZ4PBB(MxGSiz),
     $       PX3PAA(MxGSiz), PX4PAA(MxGSiz),
     $       PY3PAA(MxGSiz), PY4PAA(MxGSiz),
     $       PZ3PAA(MxGSiz), PZ4PAA(MxGSiz),
     $       XS(MxSrfs,MxGsiz,MxGsiz),
     $       YS(MxSrfs,MxGsiz,MxGsiz),
     $       ZS(MxSrfs,MxGsiz,MxGsiz),
     $       XPnt(MxGSiz,MxGSiz,MxGSiz),
     $       YPnt(MxGSiz,MxGSiz,MxGSiz),
     $       ZPnt(MxGSiz,MxGSiz,MxGSiz)
C
      EQUIVALENCE (XB(1,1),X1(1)),(YB(1,1),Y1(1)),(ZB(1,1),Z1(1)),
     $      (XB(1,2),X2(1)),(YB(1,2),Y2(1)),(ZB(1,2),Z2(1)),
     $      (XB(1,3),X3(1)),(YB(1,3),Y3(1)),(ZB(1,3),Z3(1)),
     $      (XB(1,4),X4(1)),(YB(1,4),Y4(1)),(ZB(1,4),Z4(1))
C
      EQUIVALENCE (StrB(1,1),ZtSt1(1)),(StrB(1,2),ZtSt2(1)),
     $            (StrB(1,5),ZtSt5(1)),(StrB(1,6),ZtSt6(1)),
     $            (StrB(1,11),EtSt11(1)),(StrB(1,12),EtSt12(1)),
     $            (StrB(1,15),EtSt15(1)),(StrB(1,16),EtSt16(1))
C
C Specify input and output device unit numbers for Region 1.  This is
C convenient for running the program on a PC.  For a mainframe, you will
C need to use the FORTRAN OPEN and CLOSE statements or alter the input
C to use a namelist.
C
      InNum=7
      OutNum=8
C
C
C
C
C Read in the grid control information.
C
      CALL RdGrIn(II,JJ,KK,NSurfs,SigmaXi,SigmaEt,SigmaZt,
     $            kXi1,kXi2,kEta1,kEta2,kZeta1,kZeta2,InNum)
C
C
C Set various parameters for the grid generation routines.
C
      CALL KFctrs(1,kS,k1,k2,k3,k4,kXi1,kXi2,kEta1,kEta2,
     $            kZeta1,kZeta2,MxSrfs,MxGSiz)
C
      AAL(1)=KK
      AAL(2)=KK
      AAL(3)=II
      AAL(4)=II
      BBL(1)=II
      BBL(2)=II
      BBL(3)=JJ
      BBL(4)=JJ
      SigmaAA(1)=SigmaZt
      SigmaAA(2)=SigmaZt
      SigmaAA(3)=SigmaXi
      SigmaAA(4)=SigmaXi
      SigmaBB(1)=SigmaXi
```

```
      SigmaBB(2)=SigmaXi
      SigmaBB(3)=SigmaEt
      SigmaBB(4)=SigmaEt
      NGPts(1)=KK
      NGPts(2)=KK
      NGPts(3)=II
      NGPts(4)=II
      NGPts(5)=KK
      NGPts(6)=KK
      NGPts(7)=II
      NGPts(8)=II
      NGPts(9)=II
      NGPts(10)=II
      NGPts(11)=JJ
      NGPts(12)=JJ
      NGPts(13)=II
      NGPts(14)=II
      NGPts(15)=JJ
      NGPts(16)=JJ
C
C
C Calculate the grid point spacings in the transformed domain.
C
      XiStep=1.0/(II-1)
      EtStep=1.0/(JJ-1)
      ZtStep=1.0/(KK-1)
C
C Calculate the boundary surface grid point locations for each surface.
C
      DO 40 SrfNum=1,NSurfs
C
C Read in the edge curve nodal points and form the boundary surface
C edge curves for surface SrfNum by splining.
C
         DO 30 CrvNum=1,4
            EdgNum=(SrfNum-1)*4 + CrvNum
            READ(InNum,*)Type
            IF(Type.EQ.1)THEN
               CALL RdGrPIn(NGPts(EdgNum),XB,YB,ZB,CrvNum,MxGSiz,InNum)
               CALL CalSt1(NGPts(EdgNum),XB,YB,ZB,CrvNum,EdgNum,
     $                         StrB,MxBCvs,MxGSiz)
            ELSE
               CALL RdCvIn(x,y,z,NDPts,CrvNum,Tensn,InNum,MxBPts,
     $                      StrTp,Beta1,Beta2)
               CALL PTSpln(x,y,z,s,zx,zy,zz,Diag,OfDiag,Right,NDPts,
     $                      Tensn(CrvNum),CrvNum,MxBPts)
               CALL CalSt2(EdgNum,NGPts(EdgNum),StrTp,Beta1,Beta2,
     $                         StrB,MxBCvs,MxGSiz)
               CALL EdgGPts(CrvNum,EdgNum,NGPts(EdgNum),XB,YB,ZB,StrB,
     $                        x,y,z,s,zx,zy,zz,NDPts,Tensn(CrvNum),
     $                        MxBCvs,MxBPts,MxGSiz)
            ENDIF
 30      CONTINUE
C
C
C Calculate the boundary surface edge derivative values for surface SrfNum.
C
         CALL EdgDer(PX1PAA,PX2PAA,PY1PAA,PY2PAA,PZ1PAA,PZ2PAA,
```

```
     $                 PX3PBB,PX4PBB,PY3PBB,PY4PBB,PZ3PBB,PZ4PBB,
     $                 X1,X2,X3,X4,Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,
     $                 AAL(SrfNum),BBL(SrfNum),MxGSiz)
C
C Calculate the boundary surface grid point locations for surface SrfNum.
C
         CALL TwoBnd(XS,YS,ZS,SrfNum,AAL(SrfNum),BBL(SrfNum),
     $                 SigmaBB(SrfNum),k1,k2,StrB,
     $                 h1,h2,h3,h4,X1,X2,X3,X4,
     $                 Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,PX1PBB,PX2PBB,
     $                 PY1PBB,PY2PBB,PZ1PBB,PZ2PBB,PX1PAA,PX2PAA,
     $                 PY1PAA,PY2PAA,PZ1PAA,PZ2PAA,PX3PBB,PX4PBB,
     $                 PY3PBB,PY4PBB,PZ3PBB,PZ4PBB,
     $                 MxBCvs,MxGSiz,MxSrfs)
C
         CALL ForBnd(XS,YS,ZS,SrfNum,AAL(SrfNum),BBL(SrfNum),
     $                 SigmaAA(SrfNum),SigmaBB(SrfNum),
     $                 k3,k4,StrB,
     $                 h1,h2,h3,h4,h5,h6,h7,h8,
     $                 X1,X2,X3,X4,Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,
     $                 PX1PBB,PX2PBB,PY1PBB,PY2PBB,PZ1PBB,PZ2PBB,
     $                 PX1PAA,PX2PAA,PY1PAA,PY2PAA,PZ1PAA,PZ2PAA,
     $                 PX3PBB,PX4PBB,PY3PBB,PY4PBB,PZ3PBB,PZ4PBB,
     $                 PX3PAA,PX4PAA,PY3PAA,PY4PAA,PZ3PAA,PZ4PAA,
     $                 MxBCvs,MxGSiz,MxSrfs)
C
 40   CONTINUE
C
      IF(NSurfs.EQ.2)THEN
        DO 60 SrfNum=3,4
           DO 50 CrvNum=3,4
              EdgNum=(SrfNum-1)*4 + CrvNum
              READ(InNum,*)StrTp
              IF(StrTp.NE.4)THEN
                 READ(InNum,*)Beta1
              ELSE
                 READ(InNum,*)Beta1,Beta2
              ENDIF
              CALL CalSt2(EdgNum,NGPts(EdgNum),StrTp,Beta1,Beta2,
     $                         StrB,MxBCvs,MxGSiz)
 50        CONTINUE
 60     CONTINUE
      ENDIF
C
C
C
C Calculate the interior grid point locations.
C
      CALL TwoSrf(XPnt,YPnt,ZPnt,II,JJ,KK,SigmaEt,kS,
     $              EtSt11,EtSt12,EtSt15,EtSt16,
     $              XiStep,EtStep,ZtStep,XS,YS,ZS,h1,h2,h3,h4,
     $              PXS1PE,PXS2PE,PYS1PE,PYS2PE,PZS1PE,
     $              PZS2PE,MxGSiz,MxSrfs)
C
      IF (NSurfs.EQ.4) THEN
         CALL ForSrf(XPnt,YPnt,ZPnt,II,JJ,KK,
     $                 SigmaEt,SigmaZt,kS,
     $                 EtSt11,EtSt12,EtSt15,EtSt16,
```

```
     $                ZtSt1,ZtSt2,ZtSt5,ZtSt6,
     $                XS,YS,ZS,XiStep,EtStep,ZtStep,
     $                h1,h2,h3,h4,h5,h6,h7,h8,
     $                PXS1PE,PXS2PE,PYS1PE,PYS2PE,PZS1PE,PZS2PE,
     $                PXS3Zt,PXS4Zt,PYS3Zt,PYS4Zt,PZS3Zt,PZS4Zt,
     $                MxGSiz,MxSrfs)
      ENDIF
C
      CALL PrGrid(XPnt,YPnt,ZPnt,II,JJ,KK,OutNum,MxGSiz)
C
      RETURN
      END
C
C
C=====================================================================C
C
      SUBROUTINE TwoSrf(XPnt,YPnt,ZPnt,II,JJ,KK,SigmaEt,kS,
     $                  EtSt11,EtSt12,EtSt15,EtSt16,
     $                  XiStep,EtStep,ZtStep,XS,YS,ZS,
     $                  h1,h2,h3,h4,PXS1PE,PXS2PE,PYS1PE,PYS2PE,PZS1PE,
     $                  PZS2PE,MxGSiz,MxSrfs)
C
C This SUBROUTINE calculates the grid point locations between two specified
C surfaces using the "two-boundary technique".
C
      INTEGER  i, j, k, StrXi, StrEt, StrZt, II, JJ, KK
C
      REAL  Xi, Eta, Zeta, XiNew, EtaNew, ZtaNew, LL1, LL2,
     $      PXS1Xi, PXS2Xi, PYS1Xi, PYS2Xi, PZS1Xi, PZS2Xi,
     $      PXS1Zt, PXS2Zt, PYS1Zt, PYS2Zt, PZS1Zt, PZS2Zt,
     $      BetaXi, BetaEt, BetaZt, XiStep, EtStep, ZtStep,
     $      EtSt11(MxGSiz),EtSt12(MxGSiz),
     $      EtSt15(MxGSiz),EtSt16(MxGSiz),
     $      kS(MxSrfs,MxGSiz,MxGSiz),
     $      h1(MxGSiz), h2(MxGSiz), h3(MxGSiz), h4(MxGSiz),
     $      PXS1PE(MxGSiz,MxGsiz), PXS2PE(MxGSiz,MxGsiz),
     $      PYS1PE(MxGSiz,MxGsiz), PYS2PE(MxGSiz,MxGsiz),
     $      PZS1PE(MxGSiz,MxGsiz), PZS2PE(MxGSiz,MxGsiz),
     $      XS(MxSrfs,MxGsiz,MxGsiz),
     $      YS(MxSrfs,MxGsiz,MxGsiz),
     $      ZS(MxSrfs,MxGsiz,MxGsiz),
     $      XPnt(MxGSiz,MxGsiz,MxGSiz),
     $      YPnt(MxGSiz,MxGsiz,MxGSiz),
     $      ZPnt(MxGSiz,MxGsiz,MxGSiz)
C
C Calculate the derivative values along the constant Xi/Zeta
C boundaries.
C
      PXS1Xi=(XS(1,1,2)-XS(1,1,1))/XiStep
      PXS2Xi=(XS(2,1,2)-XS(2,1,1))/XiStep
      PYS1Xi=(YS(1,1,2)-YS(1,1,1))/XiStep
      PYS2Xi=(YS(2,1,2)-YS(2,1,1))/XiStep
      PZS1Xi=(ZS(1,1,2)-ZS(1,1,1))/XiStep
      PZS2Xi=(ZS(2,1,2)-ZS(2,1,1))/XiStep
      PXS1Zt=(XS(1,2,1)-XS(1,1,1))/ZtStep
      PXS2Zt=(XS(2,2,1)-XS(2,1,1))/ZtStep
      PYS1Zt=(YS(1,2,1)-YS(1,1,1))/ZtStep
      PYS2Zt=(YS(2,2,1)-YS(2,1,1))/ZtStep
```

```
      PZS1Zt=(ZS(1,2,1)-ZS(1,1,1))/ZtStep
      PZS2Zt=(ZS(2,2,1)-ZS(2,1,1))/ZtStep
      LL1=((PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)**2
     /    +(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)**2
     /    +(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)**2)**0.5
      LL2=((PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)**2
     /    +(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)**2
     /    +(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)**2)**0.5
C
      PXS1PE(1,1)=-kS(1,1,1)*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)/LL1
      PXS2PE(1,1)=-kS(2,1,1)*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)/LL2
      PYS1PE(1,1)= kS(1,1,1)*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)/LL1
      PYS2PE(1,1)= kS(2,1,1)*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)/LL2
      PZS1PE(1,1)=-kS(1,1,1)*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)/LL1
      PZS2PE(1,1)=-kS(2,1,1)*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)/LL2
C
      PXS1Xi=(XS(1,1,II)-XS(1,1,II-1))/XiStep
      PXS2Xi=(XS(2,1,II)-XS(2,1,II-1))/XiStep
      PYS1Xi=(YS(1,1,II)-YS(1,1,II-1))/XiStep
      PYS2Xi=(YS(2,1,II)-YS(2,1,II-1))/XiStep
      PZS1Xi=(ZS(1,1,II)-ZS(1,1,II-1))/XiStep
      PZS2Xi=(ZS(2,1,II)-ZS(2,1,II-1))/XiStep
      PXS1Zt=(XS(1,2,II)-XS(1,1,II))/ZtStep
      PXS2Zt=(XS(2,2,II)-XS(2,1,II))/ZtStep
      PYS1Zt=(YS(1,2,II)-YS(1,1,II))/ZtStep
      PYS2Zt=(YS(2,2,II)-YS(2,1,II))/ZtStep
      PZS1Zt=(ZS(1,2,II)-ZS(1,1,II))/ZtStep
      PZS2Zt=(ZS(2,2,II)-ZS(2,1,II))/ZtStep
      LL1=((PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)**2
     /    +(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)**2
     /    +(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)**2)**0.5
      LL2=((PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)**2
     /    +(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)**2
     /    +(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)**2)**0.5
C
      PXS1PE(II,1)=-kS(1,II,1)*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)/LL1
      PXS2PE(II,1)=-kS(2,II,1)*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)/LL2
      PYS1PE(II,1)= kS(1,II,1)*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)/LL1
      PYS2PE(II,1)= kS(2,II,1)*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)/LL2
      PZS1PE(II,1)=-kS(1,II,1)*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)/LL1
      PZS2PE(II,1)=-kS(2,II,1)*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)/LL2
C
      DO 55 i=2,II-1
         PXS1Xi=(XS(1,1,i+1)-XS(1,1,i-1))/2/XiStep
         PXS2Xi=(XS(2,1,i+1)-XS(2,1,i-1))/2/XiStep
         PYS1Xi=(YS(1,1,i+1)-YS(1,1,i-1))/2/XiStep
         PYS2Xi=(YS(2,1,i+1)-YS(2,1,i-1))/2/XiStep
         PZS1Xi=(ZS(1,1,i+1)-ZS(1,1,i-1))/2/XiStep
         PZS2Xi=(ZS(2,1,i+1)-ZS(2,1,i-1))/2/XiStep
         PXS1Zt=(XS(1,2,i)-XS(1,1,i))/ZtStep
         PXS2Zt=(XS(2,2,i)-XS(2,1,i))/ZtStep
         PYS1Zt=(YS(1,2,i)-YS(1,1,i))/ZtStep
         PYS2Zt=(YS(2,2,i)-YS(2,1,i))/ZtStep
         PZS1Zt=(ZS(1,2,i)-ZS(1,1,i))/ZtStep
         PZS2Zt=(ZS(2,2,i)-ZS(2,1,i))/ZtStep
         LL1=((PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)**2
     /       +(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)**2
     /       +(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)**2)**0.5
```

```
        LL2=((PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)**2
     /      +(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)**2
     /      +(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)**2)**0.5
C
        PXS1PE(i,1)=-kS(1,i,1)*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)/LL1
        PXS2PE(i,1)=-kS(2,i,1)*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)/LL2
        PYS1PE(i,1)= kS(1,i,1)*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)/LL1
        PYS2PE(i,1)= kS(2,i,1)*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)/LL2
        PZS1PE(i,1)=-kS(1,i,1)*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)/LL1
        PZS2PE(i,1)=-kS(2,i,1)*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)/LL2
 55   CONTINUE
C
      DO 70 k=2,KK-1
        PXS1Xi=(XS(1,k,2)-XS(1,k,1))/XiStep
        PXS2Xi=(XS(2,k,2)-XS(2,k,1))/XiStep
        PYS1Xi=(YS(1,k,2)-YS(1,k,1))/XiStep
        PYS2Xi=(YS(2,k,2)-YS(2,k,1))/XiStep
        PZS1Xi=(ZS(1,k,2)-ZS(1,k,1))/XiStep
        PZS2Xi=(ZS(2,k,2)-ZS(2,k,1))/XiStep
        PXS1Zt=(XS(1,k+1,1)-XS(1,k-1,1))/2/ZtStep
        PXS2Zt=(XS(2,k+1,1)-XS(2,k-1,1))/2/ZtStep
        PYS1Zt=(YS(1,k+1,1)-YS(1,k-1,1))/2/ZtStep
        PYS2Zt=(YS(2,k+1,1)-YS(2,k-1,1))/2/ZtStep
        PZS1Zt=(ZS(1,k+1,1)-ZS(1,k-1,1))/2/ZtStep
        PZS2Zt=(ZS(2,k+1,1)-ZS(2,k-1,1))/2/ZtStep
        LL1=((PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)**2
     /      +(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)**2
     /      +(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)**2)**0.5
        LL2=((PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)**2
     /      +(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)**2
     /      +(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)**2)**0.5
C
        PXS1PE(1,k)=-kS(1,1,k)*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)/LL1
        PXS2PE(1,k)=-kS(2,1,k)*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)/LL2
        PYS1PE(1,k)= kS(1,1,k)*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)/LL1
        PYS2PE(1,k)= kS(2,1,k)*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)/LL2
        PZS1PE(1,k)=-kS(1,1,k)*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)/LL1
        PZS2PE(1,k)=-kS(2,1,k)*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)/LL2
C
        PXS1Xi=(XS(1,k,II)-XS(1,k,II-1))/XiStep
        PXS2Xi=(XS(2,k,II)-XS(2,k,II-1))/XiStep
        PYS1Xi=(YS(1,k,II)-YS(1,k,II-1))/XiStep
        PYS2Xi=(YS(2,k,II)-YS(2,k,II-1))/XiStep
        PZS1Xi=(ZS(1,k,II)-ZS(1,k,II-1))/XiStep
        PZS2Xi=(ZS(2,k,II)-ZS(2,k,II-1))/XiStep
        PXS1Zt=(XS(1,k+1,II)-XS(1,k-1,II))/2/ZtStep
        PXS2Zt=(XS(2,k+1,II)-XS(2,k-1,II))/2/ZtStep
        PYS1Zt=(YS(1,k+1,II)-YS(1,k-1,II))/2/ZtStep
        PYS2Zt=(YS(2,k+1,II)-YS(2,k-1,II))/2/ZtStep
        PZS1Zt=(ZS(1,k+1,II)-ZS(1,k-1,II))/2/ZtStep
        PZS2Zt=(ZS(2,k+1,II)-ZS(2,k-1,II))/2/ZtStep
        LL1=((PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)**2
     /      +(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)**2
     /      +(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)**2)**0.5
        LL2=((PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)**2
     /      +(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)**2
     /      +(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)**2)**0.5
C
```

```
         PXS1PE(II,k)=-kS(1,II,k)*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)/LL1
         PXS2PE(II,k)=-kS(2,II,k)*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)/LL2
         PYS1PE(II,k)= kS(1,II,k)*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)/LL1
         PYS2PE(II,k)= kS(2,II,k)*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)/LL2
         PZS1PE(II,k)=-kS(1,II,k)*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)/LL1
         PZS2PE(II,k)=-kS(2,II,k)*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)/LL2
C
         DO 60 i=2,II-1
            PXS1Xi=(XS(1,k,i+1)-XS(1,k,i-1))/2/XiStep
            PXS2Xi=(XS(2,k,i+1)-XS(2,k,i-1))/2/XiStep
            PYS1Xi=(YS(1,k,i+1)-YS(1,k,i-1))/2/XiStep
            PYS2Xi=(YS(2,k,i+1)-YS(2,k,i-1))/2/XiStep
            PZS1Xi=(ZS(1,k,i+1)-ZS(1,k,i-1))/2/XiStep
            PZS2Xi=(ZS(2,k,i+1)-ZS(2,k,i-1))/2/XiStep
            PXS1Zt=(XS(1,k+1,i)-XS(1,k-1,i))/2/ZtStep
            PXS2Zt=(XS(2,k+1,i)-XS(2,k-1,i))/2/ZtStep
            PYS1Zt=(YS(1,k+1,i)-YS(1,k-1,i))/2/ZtStep
            PYS2Zt=(YS(2,k+1,i)-YS(2,k-1,i))/2/ZtStep
            PZS1Zt=(ZS(1,k+1,i)-ZS(1,k-1,i))/2/ZtStep
            PZS2Zt=(ZS(2,k+1,i)-ZS(2,k-1,i))/2/ZtStep
            LL1=((PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)**2
     /           +(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)**2
     /           +(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)**2)**0.5
            LL2=((PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)**2
     /           +(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)**2
     /           +(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)**2)**0.5
C
            PXS1PE(i,k)=-kS(1,i,k)*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)/LL1
            PXS2PE(i,k)=-kS(2,i,k)*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)/LL2
            PYS1PE(i,k)= kS(1,i,k)*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)/LL1
            PYS2PE(i,k)= kS(2,i,k)*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)/LL2
            PZS1PE(i,k)=-kS(1,i,k)*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)/LL1
            PZS2PE(i,k)=-kS(2,i,k)*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)/LL2
 60      CONTINUE
 70   CONTINUE
C
      PXS1Xi=(XS(1,KK,2)-XS(1,KK,1))/XiStep
      PXS2Xi=(XS(2,KK,2)-XS(2,KK,1))/XiStep
      PYS1Xi=(YS(1,KK,2)-YS(1,KK,1))/XiStep
      PYS2Xi=(YS(2,KK,2)-YS(2,KK,1))/XiStep
      PZS1Xi=(ZS(1,KK,2)-ZS(1,KK,1))/XiStep
      PZS2Xi=(ZS(2,KK,2)-ZS(2,KK,1))/XiStep
      PXS1Zt=(XS(1,KK,1)-XS(1,KK-1,1))/ZtStep
      PXS2Zt=(XS(2,KK,1)-XS(2,KK-1,1))/ZtStep
      PYS1Zt=(YS(1,KK,1)-YS(1,KK-1,1))/ZtStep
      PYS2Zt=(YS(2,KK,1)-YS(2,KK-1,1))/ZtStep
      PZS1Zt=(ZS(1,KK,1)-ZS(1,KK-1,1))/ZtStep
      PZS2Zt=(ZS(2,KK,1)-ZS(2,KK-1,1))/ZtStep
      LL1=((PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)**2
     /     +(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)**2
     /     +(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)**2)**0.5
      LL2=((PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)**2
     /     +(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)**2
     /     +(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)**2)**0.5
C
      PXS1PE(1,KK)=-kS(1,1,KK)*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)/LL1
      PXS2PE(1,KK)=-kS(2,1,KK)*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)/LL2
      PYS1PE(1,KK)= kS(1,1,KK)*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)/LL1
```

```
      PYS2PE(1,KK)= kS(2,1,KK)*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)/LL2
      PZS1PE(1,KK)=-kS(1,1,KK)*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)/LL1
      PZS2PE(1,KK)=-kS(2,1,KK)*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)/LL2
C
      PXS1Xi=(XS(1,KK,II)-XS(1,KK,II-1))/XiStep
      PXS2Xi=(XS(2,KK,II)-XS(2,KK,II-1))/XiStep
      PYS1Xi=(YS(1,KK,II)-YS(1,KK,II-1))/XiStep
      PYS2Xi=(YS(2,KK,II)-YS(2,KK,II-1))/XiStep
      PZS1Xi=(ZS(1,KK,II)-ZS(1,KK,II-1))/XiStep
      PZS2Xi=(ZS(2,KK,II)-ZS(2,KK,II-1))/XiStep
      PXS1Zt=(XS(1,KK,II)-XS(1,KK-1,II))/ZtStep
      PXS2Zt=(XS(2,KK,II)-XS(2,KK-1,II))/ZtStep
      PYS1Zt=(YS(1,KK,II)-YS(1,KK-1,II))/ZtStep
      PYS2Zt=(YS(2,KK,II)-YS(2,KK-1,II))/ZtStep
      PZS1Zt=(ZS(1,KK,II)-ZS(1,KK-1,II))/ZtStep
      PZS2Zt=(ZS(2,KK,II)-ZS(2,KK-1,II))/ZtStep
      LL1=((PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)**2
     /    +(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)**2
     /    +(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)**2)**0.5
      LL2=((PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)**2
     /    +(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)**2
     /    +(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)**2)**0.5
C
      PXS1PE(II,KK)=-kS(1,II,KK)*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)/LL1
      PXS2PE(II,KK)=-kS(2,II,KK)*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)/LL2
      PYS1PE(II,KK)= kS(1,II,KK)*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)/LL1
      PYS2PE(II,KK)= kS(2,II,KK)*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)/LL2
      PZS1PE(II,KK)=-kS(1,II,KK)*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)/LL1
      PZS2PE(II,KK)=-kS(2,II,KK)*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)/LL2
C
      DO 75 i=2,II-1
         PXS1Xi=(XS(1,KK,i+1)-XS(1,KK,i-1))/2/XiStep
         PXS2Xi=(XS(2,KK,i+1)-XS(2,KK,i-1))/2/XiStep
         PYS1Xi=(YS(1,KK,i+1)-YS(1,KK,i-1))/2/XiStep
         PYS2Xi=(YS(2,KK,i+1)-YS(2,KK,i-1))/2/XiStep
         PZS1Xi=(ZS(1,KK,i+1)-ZS(1,KK,i-1))/2/XiStep
         PZS2Xi=(ZS(2,KK,i+1)-ZS(2,KK,i-1))/2/XiStep
         PXS1Zt=(XS(1,KK,i)-XS(1,KK-1,i))/ZtStep
         PXS2Zt=(XS(2,KK,i)-XS(2,KK-1,i))/ZtStep
         PYS1Zt=(YS(1,KK,i)-YS(1,KK-1,i))/ZtStep
         PYS2Zt=(YS(2,KK,i)-YS(2,KK-1,i))/ZtStep
         PZS1Zt=(ZS(1,KK,i)-ZS(1,KK-1,i))/ZtStep
         PZS2Zt=(ZS(2,KK,i)-ZS(2,KK-1,i))/ZtStep
         LL1=((PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)**2
     /       +(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)**2
     /       +(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)**2)**0.5
         LL2=((PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)**2
     /       +(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)**2
     /       +(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)**2)**0.5
C
         PXS1PE(i,KK)=-kS(1,i,KK)*(PYS1Xi*PZS1Zt-PZS1Xi*PYS1Zt)/LL1
         PXS2PE(i,KK)=-kS(2,i,KK)*(PYS2Xi*PZS2Zt-PZS2Xi*PYS2Zt)/LL2
         PYS1PE(i,KK)= kS(1,i,KK)*(PXS1Xi*PZS1Zt-PZS1Xi*PXS1Zt)/LL1
         PYS2PE(i,KK)= kS(2,i,KK)*(PXS2Xi*PZS2Zt-PZS2Xi*PXS2Zt)/LL2
         PZS1PE(i,KK)=-kS(1,i,KK)*(PXS1Xi*PYS1Zt-PYS1Xi*PXS1Zt)/LL1
         PZS2PE(i,KK)=-kS(2,i,KK)*(PXS2Xi*PYS2Zt-PYS2Xi*PXS2Zt)/LL2
 75   CONTINUE
C
```

```
C Calculate the interior grid point locations.
C
      DO 100 k=1,KK
         Zeta=(k-1.)/(KK-1.)
         DO 90 i=1,II
            Xi=(i-1.)/(II-1.)
            DO 80 j=1,JJ
               EtaNew=(EtSt11(j)*(1.-Xi)+EtSt12(j)*Xi)*(1.-Zeta)
     $                +(EtSt15(j)*(1.-Xi)+EtSt16(j)*Xi)*Zeta
               CALL FindHs(h1(j),h2(j),h3(j),h4(j),EtaNew,SigmaEt)
               XPnt(i,j,k)=h1(j)
     $                     *XS(1,k,i)+h2(j)*XS(2,k,i)
     $                     +h3(j)*PXS1PE(i,k)
     $                     +h4(j)*PXS2PE(i,k)
               YPnt(i,j,k)=h1(j)
     $                     *YS(1,k,i)+h2(j)*YS(2,k,i)
     $                     +h3(j)*PYS1PE(i,k)
     $                     +h4(j)*PYS2PE(i,k)
               ZPnt(i,j,k)=h1(j)
     $                     *ZS(1,k,i)+h2(j)*ZS(2,k,i)
     $                     +h3(j)*PZS1PE(i,k)
     $                     +h4(j)*PZS2PE(i,k)
 80         CONTINUE
 90      CONTINUE
 100  CONTINUE
C
      RETURN
      END
C
C====================================================================C
C
      SUBROUTINE ForSrf(XPnt,YPnt,ZPnt,II,JJ,KK,SigmaEt,SigmaZt,kS,
     $                  EtSt11,EtSt12,EtSt15,EtSt16,
     $                  ZtSt1,ZtSt2,ZtSt5,ZtSt6,
     $                  XS,YS,ZS,XiStep,EtStep,ZtStep,
     $                  h1,h2,h3,h4,h5,h6,h7,h8,
     $                  PXS1PE,PXS2PE,PYS1PE,PYS2PE,PZS1PE,PZS2PE,
     $                  PXS3Zt,PXS4Zt,PYS3Zt,PYS4Zt,PZS3Zt,PZS4Zt,
     $                  MxGSiz,MxSrfs)
C
C
C This SUBROUTINE adjusts the grid so that the other two surfaces of the
C region are mapped correctly using the "four-boundary technique".
C
      INTEGER  i, j, k, StrXi, StrEt, StrZt, II, JJ, KK
C
      REAL  Xi, Eta, Zeta, XiNew, EtaNew, ZtaNew, LL3, LL4,
     $      h1(MxGSiz), h2(MxGSiz), h3(MxGSiz), h4(MxGSiz),
     $      h5(MxGSiz), h6(MxGSiz), h7(MxGSiz), h8(MxGSiz),
     $      PXS3Xi,  PXS4Xi, PYS3Xi,  PYS4Xi, PZS3Xi,  PZS4Xi,
     $      PXS3PE, PXS4PE, PYS3PE, PYS4PE, PZS3PE, PZS4PE,
     $      P2X00, P2X01, P2X10, P2X11, P2Y00, P2Y01, P2Y10, P2Y11,
     $      P2Z00, P2Z01, P2Z10, P2Z11
      REAL  BetaXi, BetaEt, BetaZt, XiStep, EtStep, ZtStep,
     $      EtSt11(MxGSiz),EtSt12(MxGSiz),
     $      EtSt15(MxGSiz),EtSt16(MxGSiz),
     $      ZtSt1(MxGSiz),ZtSt2(MxGSiz),
     $      ZtSt5(MxGSiz),ZtSt6(MxGSiz),
```

```
     $       kS(MxSrfs,MxGSiz,MxGSiz),
     $       PXS1PE(MxGSiz,MxGsiz), PXS2PE(MxGSiz,MxGsiz),
     $       PXS3Zt(MxGSiz,MxGsiz), PXS4Zt(MxGSiz,MxGsiz),
     $       PYS1PE(MxGSiz,MxGsiz), PYS2PE(MxGSiz,MxGsiz),
     $       PYS3Zt(MxGSiz,MxGsiz), PYS4Zt(MxGSiz,MxGsiz),
     $       PZS1PE(MxGSiz,MxGsiz), PZS2PE(MxGSiz,MxGsiz),
     $       PZS3Zt(MxGSiz,MxGsiz), PZS4Zt(MxGSiz,MxGsiz),
     $       XS(MxSrfs,MxGSiz,MxGsiz),
     $       YS(MxSrfs,MxGSiz,MxGsiz),
     $       ZS(MxSrfs,MxGSiz,MxGsiz),
     $       XPnt(MxGSiz,MxGsiz,MxGSiz),
     $       YPnt(MxGSiz,MxGsiz,MxGSiz),
     $       ZPnt(MxGSiz,MxGsiz,MxGSiz)
C
C
C Calculate the derivative values along the constant Xi/Eta
C boundaries.
C
      PXS3Xi=(XS(3,2,1)-XS(3,1,1))/XiStep
      PXS4Xi=(XS(4,2,1)-XS(4,1,1))/XiStep
      PYS3Xi=(YS(3,2,1)-YS(3,1,1))/XiStep
      PYS4Xi=(YS(4,2,1)-YS(4,1,1))/XiStep
      PZS3Xi=(ZS(3,2,1)-ZS(3,1,1))/XiStep
      PZS4Xi=(ZS(4,2,1)-ZS(4,1,1))/XiStep
      PXS3PE=(XS(3,1,2)-XS(3,1,1))/EtStep
      PXS4PE=(XS(4,1,2)-XS(4,1,1))/EtStep
      PYS3PE=(YS(3,1,2)-YS(3,1,1))/EtStep
      PYS4PE=(YS(4,1,2)-YS(4,1,1))/EtStep
      PZS3PE=(ZS(3,1,2)-ZS(3,1,1))/EtStep
      PZS4PE=(ZS(4,1,2)-ZS(4,1,1))/EtStep
      LL3=((PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)**2
     /    +(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)**2
     /    +(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)**2)**0.5
      LL4=((PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)**2
     /    +(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)**2
     /    +(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)**2)**0.5
C
      PXS3Zt(1,1)=-kS(3,1,1)*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)/LL3
      PXS4Zt(1,1)=-kS(4,1,1)*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)/LL4
      PYS3Zt(1,1)= kS(3,1,1)*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)/LL3
      PYS4Zt(1,1)= kS(4,1,1)*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)/LL4
      PZS3Zt(1,1)=-kS(3,1,1)*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)/LL3
      PZS4Zt(1,1)=-kS(4,1,1)*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)/LL4
C
      PXS3Xi=(XS(3,II,1)-XS(3,II-1,1))/XiStep
      PXS4Xi=(XS(4,II,1)-XS(4,II-1,1))/XiStep
      PYS3Xi=(YS(3,II,1)-YS(3,II-1,1))/XiStep
      PYS4Xi=(YS(4,II,1)-YS(4,II-1,1))/XiStep
      PZS3Xi=(ZS(3,II,1)-ZS(3,II-1,1))/XiStep
      PZS4Xi=(ZS(4,II,1)-ZS(4,II-1,1))/XiStep
      PXS3PE=(XS(3,II,2)-XS(3,II,1))/EtStep
      PXS4PE=(XS(4,II,2)-XS(4,II,1))/EtStep
      PYS3PE=(YS(3,II,2)-YS(3,II,1))/EtStep
      PYS4PE=(YS(4,II,2)-YS(4,II,1))/EtStep
      PZS3PE=(ZS(3,II,2)-ZS(3,II,1))/EtStep
      PZS4PE=(ZS(4,II,2)-ZS(4,II,1))/EtStep
      LL3=((PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)**2
     /    +(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)**2
```

```
     /     +(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)**2)**0.5
      LL4=((PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)**2
     /     +(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)**2
     /     +(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)**2)**0.5
C
      PXS3Zt(II,1)=-kS(3,II,1)*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)/LL3
      PXS4Zt(II,1)=-kS(4,II,1)*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)/LL4
      PYS3Zt(II,1)= kS(3,II,1)*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)/LL3
      PYS4Zt(II,1)= kS(4,II,1)*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)/LL4
      PZS3Zt(II,1)=-kS(3,II,1)*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)/LL3
      PZS4Zt(II,1)=-kS(4,II,1)*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)/LL4
C
      DO 45 i=2,II-1
         PXS3Xi=(XS(3,i+1,1)-XS(3,i-1,1))/2/XiStep
         PXS4Xi=(XS(4,i+1,1)-XS(4,i-1,1))/2/XiStep
         PYS3Xi=(YS(3,i+1,1)-YS(3,i-1,1))/2/XiStep
         PYS4Xi=(YS(4,i+1,1)-YS(4,i-1,1))/2/XiStep
         PZS3Xi=(ZS(3,i+1,1)-ZS(3,i-1,1))/2/XiStep
         PZS4Xi=(ZS(4,i+1,1)-ZS(4,i-1,1))/2/XiStep
         PXS3PE=(XS(3,i,2)-XS(3,i,1))/EtStep
         PXS4PE=(XS(4,i,2)-XS(4,i,1))/EtStep
         PYS3PE=(YS(3,i,2)-YS(3,i,1))/EtStep
         PYS4PE=(YS(4,i,2)-YS(4,i,1))/EtStep
         PZS3PE=(ZS(3,i,2)-ZS(3,i,1))/EtStep
         PZS4PE=(ZS(4,i,2)-ZS(4,i,1))/EtStep
         LL3=((PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)**2
     /        +(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)**2
     /        +(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)**2)**0.5
         LL4=((PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)**2
     /        +(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)**2
     /        +(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)**2)**0.5
C
         PXS3Zt(i,1)=-kS(3,i,1)*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)/LL3
         PXS4Zt(i,1)=-kS(4,i,1)*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)/LL4
         PYS3Zt(i,1)= kS(3,i,1)*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)/LL3
         PYS4Zt(i,1)= kS(4,i,1)*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)/LL4
         PZS3Zt(i,1)=-kS(3,i,1)*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)/LL3
         PZS4Zt(i,1)=-kS(4,i,1)*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)/LL4
 45   CONTINUE
C
      DO 60 j=2,JJ-1
         PXS3Xi=(XS(3,2,j)-XS(3,1,j))/XiStep
         PXS4Xi=(XS(4,2,j)-XS(4,1,j))/XiStep
         PYS3Xi=(YS(3,2,j)-YS(3,1,j))/XiStep
         PYS4Xi=(YS(4,2,j)-YS(4,1,j))/XiStep
         PZS3Xi=(ZS(3,2,j)-ZS(3,1,j))/XiStep
         PZS4Xi=(ZS(4,2,j)-ZS(4,1,j))/XiStep
         PXS3PE=(XS(3,1,j+1)-XS(3,1,j-1))/2/EtStep
         PXS4PE=(XS(4,1,j+1)-XS(4,1,j-1))/2/EtStep
         PYS3PE=(YS(3,1,j+1)-YS(3,1,j-1))/2/EtStep
         PYS4PE=(YS(4,1,j+1)-YS(4,1,j-1))/2/EtStep
         PZS3PE=(ZS(3,1,j+1)-ZS(3,1,j-1))/2/EtStep
         PZS4PE=(ZS(4,1,j+1)-ZS(4,1,j-1))/2/EtStep
         LL3=((PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)**2
     /        +(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)**2
     /        +(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)**2)**0.5
         LL4=((PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)**2
     /        +(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)**2
```

```
     /        +(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)**2)**0.5
C
          PXS3Zt(1,j)=-kS(3,1,j)*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)/LL3
          PXS4Zt(1,j)=-kS(4,1,j)*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)/LL4
          PYS3Zt(1,j)= kS(3,1,j)*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)/LL3
          PYS4Zt(1,j)= kS(4,1,j)*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)/LL4
          PZS3Zt(1,j)=-kS(3,1,j)*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)/LL3
          PZS4Zt(1,j)=-kS(4,1,j)*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)/LL4
C
          PXS3Xi=(XS(3,II,j)-XS(3,II-1,j))/XiStep
          PXS4Xi=(XS(4,II,j)-XS(4,II-1,j))/XiStep
          PYS3Xi=(YS(3,II,j)-YS(3,II-1,j))/XiStep
          PYS4Xi=(YS(4,II,j)-YS(4,II-1,j))/XiStep
          PZS3Xi=(ZS(3,II,j)-ZS(3,II-1,j))/XiStep
          PZS4Xi=(ZS(4,II,j)-ZS(4,II-1,j))/XiStep
          PXS3PE=(XS(3,II,j+1)-XS(3,II,j-1))/2/EtStep
          PXS4PE=(XS(4,II,j+1)-XS(4,II,j-1))/2/EtStep
          PYS3PE=(YS(3,II,j+1)-YS(3,II,j-1))/2/EtStep
          PYS4PE=(YS(4,II,j+1)-YS(4,II,j-1))/2/EtStep
          PZS3PE=(ZS(3,II,j+1)-ZS(3,II,j-1))/2/EtStep
          PZS4PE=(ZS(4,II,j+1)-ZS(4,II,j-1))/2/EtStep
          LL3=((PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)**2
     /        +(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)**2
     /        +(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)**2)**0.5
          LL4=((PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)**2
     /        +(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)**2
     /        +(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)**2)**0.5
C
          PXS3Zt(II,j)=-kS(3,II,j)*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)/LL3
          PXS4Zt(II,j)=-kS(4,II,j)*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)/LL4
          PYS3Zt(II,j)= kS(3,II,j)*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)/LL3
          PYS4Zt(II,j)= kS(4,II,j)*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)/LL4
          PZS3Zt(II,j)=-kS(3,II,j)*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)/LL3
          PZS4Zt(II,j)=-kS(4,II,j)*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)/LL4
C
          DO 50 i=2,II-1
             PXS3Xi=(XS(3,i+1,j)-XS(3,i-1,j))/2/XiStep
             PXS4Xi=(XS(4,i+1,j)-XS(4,i-1,j))/2/XiStep
             PYS3Xi=(YS(3,i+1,j)-YS(3,i-1,j))/2/XiStep
             PYS4Xi=(YS(4,i+1,j)-YS(4,i-1,j))/2/XiStep
             PZS3Xi=(ZS(3,i+1,j)-ZS(3,i-1,j))/2/XiStep
             PZS4Xi=(ZS(4,i+1,j)-ZS(4,i-1,j))/2/XiStep
             PXS3PE=(XS(3,i,j+1)-XS(3,i,j-1))/2/EtStep
             PXS4PE=(XS(4,i,j+1)-XS(4,i,j-1))/2/EtStep
             PYS3PE=(YS(3,i,j+1)-YS(3,i,j-1))/2/EtStep
             PYS4PE=(YS(4,i,j+1)-YS(4,i,j-1))/2/EtStep
             PZS3PE=(ZS(3,i,j+1)-ZS(3,i,j-1))/2/EtStep
             PZS4PE=(ZS(4,i,j+1)-ZS(4,i,j-1))/2/EtStep
             LL3=((PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)**2
     /           +(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)**2
     /           +(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)**2)**0.5
             LL4=((PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)**2
     /           +(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)**2
     /           +(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)**2)**0.5
C
             PXS3Zt(i,j)=-kS(3,i,j)*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)/LL3
             PXS4Zt(i,j)=-kS(4,i,j)*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)/LL4
             PYS3Zt(i,j)= kS(3,i,j)*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)/LL3
```

```
          PYS4Zt(i,j)= kS(4,i,j)*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)/LL4
          PZS3Zt(i,j)=-kS(3,i,j)*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)/LL3
          PZS4Zt(i,j)=-kS(4,i,j)*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)/LL4
 50     CONTINUE
 60   CONTINUE
C
      PXS3Xi=(XS(3,2,JJ)-XS(3,1,JJ))/XiStep
      PXS4Xi=(XS(4,2,JJ)-XS(4,1,JJ))/XiStep
      PYS3Xi=(YS(3,2,JJ)-YS(3,1,JJ))/XiStep
      PYS4Xi=(YS(4,2,JJ)-YS(4,1,JJ))/XiStep
      PZS3Xi=(ZS(3,2,JJ)-ZS(3,1,JJ))/XiStep
      PZS4Xi=(ZS(4,2,JJ)-ZS(4,1,JJ))/XiStep
      PXS3PE=(XS(3,1,JJ)-XS(3,1,JJ-1))/EtStep
      PXS4PE=(XS(4,1,JJ)-XS(4,1,JJ-1))/EtStep
      PYS3PE=(YS(3,1,JJ)-YS(3,1,JJ-1))/EtStep
      PYS4PE=(YS(4,1,JJ)-YS(4,1,JJ-1))/EtStep
      PZS3PE=(ZS(3,1,JJ)-ZS(3,1,JJ-1))/EtStep
      PZS4PE=(ZS(4,1,JJ)-ZS(4,1,JJ-1))/EtStep
      LL3=((PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)**2
     /    +(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)**2
     /    +(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)**2)**0.5
      LL4=((PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)**2
     /    +(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)**2
     /    +(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)**2)**0.5
C
      PXS3Zt(1,JJ)=-kS(3,1,JJ)*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)/LL3
      PXS4Zt(1,JJ)=-kS(4,1,JJ)*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)/LL4
      PYS3Zt(1,JJ)= kS(3,1,JJ)*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)/LL3
      PYS4Zt(1,JJ)= kS(4,1,JJ)*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)/LL4
      PZS3Zt(1,JJ)=-kS(3,1,JJ)*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)/LL3
      PZS4Zt(1,JJ)=-kS(4,1,JJ)*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)/LL4
C
      PXS3Xi=(XS(3,II,JJ)-XS(3,II-1,JJ))/XiStep
      PXS4Xi=(XS(4,II,JJ)-XS(4,II-1,JJ))/XiStep
      PYS3Xi=(YS(3,II,JJ)-YS(3,II-1,JJ))/XiStep
      PYS4Xi=(YS(4,II,JJ)-YS(4,II-1,JJ))/XiStep
      PZS3Xi=(ZS(3,II,JJ)-ZS(3,II-1,JJ))/XiStep
      PZS4Xi=(ZS(4,II,JJ)-ZS(4,II-1,JJ))/XiStep
      PXS3PE=(XS(3,II,JJ)-XS(3,II,JJ-1))/EtStep
      PXS4PE=(XS(4,II,JJ)-XS(4,II,JJ-1))/EtStep
      PYS3PE=(YS(3,II,JJ)-YS(3,II,JJ-1))/EtStep
      PYS4PE=(YS(4,II,JJ)-YS(4,II,JJ-1))/EtStep
      PZS3PE=(ZS(3,II,JJ)-ZS(3,II,JJ-1))/EtStep
      PZS4PE=(ZS(4,II,JJ)-ZS(4,II,JJ-1))/EtStep
      LL3=((PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)**2
     /    +(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)**2
     /    +(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)**2)**0.5
      LL4=((PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)**2
     /    +(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)**2
     /    +(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)**2)**0.5
C
      PXS3Zt(II,JJ)=-kS(3,II,JJ)*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)/LL3
      PXS4Zt(II,JJ)=-kS(4,II,JJ)*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)/LL4
      PYS3Zt(II,JJ)= kS(3,II,JJ)*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)/LL3
      PYS4Zt(II,JJ)= kS(4,II,JJ)*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)/LL4
      PZS3Zt(II,JJ)=-kS(3,II,JJ)*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)/LL3
      PZS4Zt(II,JJ)=-kS(4,II,JJ)*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)/LL4
C
```

```
      DO 65 i=2,II-1
         PXS3Xi=(XS(3,i+1,JJ)-XS(3,i-1,JJ))/2/XiStep
         PXS4Xi=(XS(4,i+1,JJ)-XS(4,i-1,JJ))/2/XiStep
         PYS3Xi=(YS(3,i+1,JJ)-YS(3,i-1,JJ))/2/XiStep
         PYS4Xi=(YS(4,i+1,JJ)-YS(4,i-1,JJ))/2/XiStep
         PZS3Xi=(ZS(3,i+1,JJ)-ZS(3,i-1,JJ))/2/XiStep
         PZS4Xi=(ZS(4,i+1,JJ)-ZS(4,i-1,JJ))/2/XiStep
         PXS3PE=(XS(3,i,JJ)-XS(3,i,JJ-1))/EtStep
         PXS4PE=(XS(4,i,JJ)-XS(4,i,JJ-1))/EtStep
         PYS3PE=(YS(3,i,JJ)-YS(3,i,JJ-1))/EtStep
         PYS4PE=(YS(4,i,JJ)-YS(4,i,JJ-1))/EtStep
         PZS3PE=(ZS(3,i,JJ)-ZS(3,i,JJ-1))/EtStep
         LL3=((PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)**2
     /          +(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)**2
     /          +(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)**2)**0.5
         LL4=((PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)**2
     /          +(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)**2
     /          +(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)**2)**0.5
C
         PXS3Zt(i,JJ)=-kS(3,i,JJ)*(PYS3Xi*PZS3PE-PZS3Xi*PYS3PE)/LL3
         PXS4Zt(i,JJ)=-kS(4,i,JJ)*(PYS4Xi*PZS4PE-PZS4Xi*PYS4PE)/LL4
         PYS3Zt(i,JJ)= kS(3,i,JJ)*(PXS3Xi*PZS3PE-PZS3Xi*PXS3PE)/LL3
         PYS4Zt(i,JJ)= kS(4,i,JJ)*(PXS4Xi*PZS4PE-PZS4Xi*PXS4PE)/LL4
         PZS3Zt(i,JJ)=-kS(3,i,JJ)*(PXS3Xi*PYS3PE-PYS3Xi*PXS3PE)/LL3
         PZS4Zt(i,JJ)=-kS(4,i,JJ)*(PXS4Xi*PYS4PE-PYS4Xi*PXS4PE)/LL4
 65   CONTINUE
C
      P2X00=0.0
      P2X10=0.0
      P2X01=0.0
      P2X11=0.0
      P2Y00=0.0
      P2Y10=0.0
      P2Y01=0.0
      P2Y11=0.0
      P2Z00=0.0
      P2Z10=0.0
      P2Z01=0.0
      P2Z11=0.0
C
C
C Calculate the grid point locations everywhere.
C
      DO 90 k=1,KK
         Zeta=(k-1.)/(KK-1.)
         DO 80 i=1,II
            Xi=(i-1.)/(II-1.)
            DO 70 j=1,JJ
               Eta=(j-1.)/(JJ-1.)
               EtaNew=(EtSt11(j)*(1.-Xi)+EtSt12(j)*Xi)*(1.-Zeta)
     $                  +(EtSt15(j)*(1.-Xi)+EtSt16(j)*Xi)*Zeta
               ZetaNew=(ZtSt1(k)*(1.-Xi)+ZtSt2(k)*Xi)*(1.-Eta)
     $                   +(ZtSt5(k)*(1.-Xi)+ZtSt6(k)*Xi)*Eta
               CALL FindHs(h1(j),h2(j),h3(j),h4(j),EtaNew,SigmaEt)
               CALL FindHs(h5(k),h6(k),h7(k),h8(k),ZetaNew,SigmaZt)
               XPnt(i,j,k)=XPnt(i,j,k)
     $              +(XS(3,i,j)-h1(j)*XS(1,1,i)
     $                          -h2(j)*XS(2,1,i)
```

```
     $                          -h3(j)*PXS1PE(i,1)
     $                          -h4(j)*PXS2PE(i,1))*h5(k)
     $                +(XS(4,i,j)-h1(j)*XS(1,KK,i)
     $                          -h2(j)*XS(2,KK,i)
     $                          -h3(j)*PXS1PE(i,KK)
     $                          -h4(j)*PXS2PE(i,KK))*h6(k)
     $                +(PXS3Zt(i,j)-(h1(j)*PXS3Zt(i,1)
     $                             +h2(j)*PXS3Zt(i,JJ)
     $                             +h3(j)*P2X00+h4(j)*P2X01))*h7(k)
     $                +(PXS4Zt(i,j)-(h1(j)*PXS4Zt(i,1)
     $                             +h2(j)*PXS4Zt(i,JJ)
     $                             +h3(j)*P2X10+h4(j)*P2X11))*h8(k)
              YPnt(i,j,k)=YPnt(i,j,k)
     $                +(YS(3,i,j)-h1(j)*YS(1,1,i)
     $                          -h2(j)*YS(2,1,i)
     $                          -h3(j)*PYS1PE(i,1)
     $                          -h4(j)*PYS2PE(i,1))*h5(k)
     $                +(YS(4,i,j)-h1(j)*YS(1,KK,i)
     $                          -h2(j)*YS(2,KK,i)
     $                          -h3(j)*PYS1PE(i,KK)
     $                          -h4(j)*PYS2PE(i,KK))*h6(k)
     $                +(PYS3Zt(i,j)-(h1(j)*PYS3Zt(i,1)
     $                             +h2(j)*PYS3Zt(i,JJ)
     $                             +h3(j)*P2Y00+h4(j)*P2Y01))*h7(k)
     $                +(PYS4Zt(i,j)-(h1(j)*PYS4Zt(i,1)
     $                             +h2(j)*PYS4Zt(i,JJ)
     $                             +h3(j)*P2Y10+h4(j)*P2Y11))*h8(k)
              ZPnt(i,j,k)=ZPnt(i,j,k)
     $                +(ZS(3,i,j)-h1(j)*ZS(1,1,i)
     $                          -h2(j)*ZS(2,1,i)
     $                          -h3(j)*PZS1PE(i,1)
     $                          -h4(j)*PZS2PE(i,1))*h5(k)
     $                +(ZS(4,i,j)-h1(j)*ZS(1,KK,i)
     $                          -h2(j)*ZS(2,KK,i)
     $                          -h3(j)*PZS1PE(i,KK)
     $                          -h4(j)*PZS2PE(i,KK))*h6(k)
     $                +(PZS3Zt(i,j)-(h1(j)*PZS3Zt(i,1)
     $                             +h2(j)*PZS3Zt(i,JJ)
     $                             +h3(j)*P2Z00+h4(j)*P2Z01))*h7(k)
     $                +(PZS4Zt(i,j)-(h1(j)*PZS4Zt(i,1)
     $                             +h2(j)*PZS4Zt(i,JJ)
     $                             +h3(j)*P2Z10+h4(j)*P2Z11))*h8(k)
 70          CONTINUE
 80        CONTINUE
 90    CONTINUE
C
       RETURN
       END
C
C===================================================================C
C
       SUBROUTINE PrGrid (XPnt,YPnt,ZPnt,II,JJ,KK,OutNum,MxGSiz)
C
C This SUBROUTINE prints (to output) the grid point x, y, and z coordinates.
C
       INTEGER  i, j, k, II, JJ, KK, OutNum
C
       REAL  XPnt(MxGSiz,MxGsiz,MxGSiz),
```

```
     $       YPnt(MxGSiz,MxGsiz,MxGSiz),
     $       ZPnt(MxGSiz,MxGsiz,MxGSiz)
C
      WRITE(OutNum,*) II
      WRITE(OutNum,*) JJ
      WRITE(OutNum,*) KK
C
      DO 30 i=1,II
         DO 20 j=1,JJ
            DO 10 k=1,KK
               WRITE(OutNum,35) XPnt(i,j,k),YPnt(i,j,k),ZPnt(i,j,k)
 10         CONTINUE
 20      CONTINUE
 30   CONTINUE
C
 35   FORMAT(1X,F10.6,3x,F10.6,3X,F10.6)
C
      RETURN
      END
C
C=====================================================================C
C
      SUBROUTINE RdGrIn(II,JJ,KK,NSurfs,SigmaXi,SigmaEt,SigmaZt,
     $                  kXi1,kXi2,kEta1,kEta2,kZeta1,kZeta2,InNum)
C
C This SUBROUTINE reads in the desired grid information for grid control.
C
      INTEGER  StrXi, StrEt, StrZt, InNum, II, JJ, KK
C
      REAL  kXi1, kXi2, kEta1, kEta2,  kZeta1, kZeta2,
     $      BetaXi, BetaEt, BetaZt, SigmaXi, SigmaEt, SigmaZt
C
      READ(InNum,*) NSurfs
C
      READ(InNum,*) II
      READ(InNum,*) JJ
      READ(InNum,*) KK
C
      READ(InNum,*) SigmaXi
      READ(InNum,*) SigmaEt
      READ(InNum,*) SigmaZt
C
      READ(InNum,*) kXi1
      READ(InNum,*) kXi2
      READ(InNum,*) kEta1
      READ(InNum,*) kEta2
      READ(InNum,*) kZeta1
      READ(InNum,*) kZeta2
C
      RETURN
      END
C
C=====================================================================C
C
      SUBROUTINE RdCvIn (x,y,z,NDPts,CrvNum,Tensn,InNum,MxBPts,
     $                   StrTp,Beta1,Beta2)
C
C This SUBROUTINE reads in the information concerning discrete points on
```

```
C the boundaries.  This information is used for generating spline-fitted
C boundary approximation curves.
C
      INTEGER  CrvNum, i, NDPts(4), InNum, StrTp
C
      REAL  x(4,MxBPts), y(4,MxBPts),
     $      z(4,MxBPts), Tensn(4)
C
      READ(InNum,*) Tensn(CrvNum)
      READ(InNum,*) NDPts(CrvNum)
C
      DO 10 i=1,NDPts(CrvNum)
         READ(InNum,*) x(CrvNum,i), y(CrvNum,i), z(CrvNum,i)
 10   CONTINUE
C
      READ(InNum,*)StrTp
      IF(StrTp.NE.4)THEN
         READ(InNum,*)Beta1
      ELSE
         READ(InNum,*)Beta1,Beta2
      ENDIF
C
      RETURN
      END
C
C=========================================================================
C
      SUBROUTINE CalcS (x,y,z,s,NDPts,CrvNum,MxBPts)
C
C This SUBROUTINE calculates the spline parameter, s, as an approximate
C arc length.
C
      INTEGER  NDPts(4), CrvNum, i
C
      REAL  x(4,MxBPts), y(4,MxBPts),
     $      z(4,MxBPts), s(4,MxBPts)
C
      s(CrvNum,1)=0.0
C
      DO 10 i=2,NDPts(CrvNum)
       s(CrvNum,i)=s(CrvNum,i-1)
     $                   +SQRT( (x(CrvNum,i)-x(CrvNum,i-1))**2
     $                         +(y(CrvNum,i)-y(CrvNum,i-1))**2
     $                         +(z(CrvNum,i)-z(CrvNum,i-1))**2)
 10   CONTINUE
C
      RETURN
      END
C
C=========================================================================
C
      SUBROUTINE SplMat (Diag,OfDiag,Right,w,s,NDPts,T,CrvNum,MxBPts)
C
C This SUBROUTINE forms the parametric tension spline matrix for a
C particular boundary curve data set.
C
      INTEGER  i, NDPts(4), CrvNum
C
```

```
      REAL Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts),
     $      w(4,MxBPts), s(4,MxBPts), T, h, hm
C
      Diag(1)=1.0
      OfDiag(1)=0.0
      Right(1)=0.0
C
      DO 10 i=2,NDPts(CrvNum)-1
         h=s(CrvNum,i+1)-s(CrvNum,i)
         hm=s(CrvNum,i)-s(CrvNum,i-1)
         Diag(i)=(T*COSH(T*hm)/SINH(T*hm)-1/hm+T*COSH(T*h)/SINH(T*h)
     $              -1/h)/T**2
         OfDiag(i)=(1/h-T/SINH(T*h))/T**2
         Right(i)= (w(CrvNum,i+1)-w(CrvNum,i))/h
     $              -(w(CrvNum,i)-w(CrvNum,i-1))/hm
 10   CONTINUE
C
      Diag(NDPts(CrvNum))=1.0
      OfDiag(NDPts(CrvNum)-1)=0.0
      Right(NDPts(CrvNum))=0.0
C
      RETURN
      END
C
C=====================================================================
C
      SUBROUTINE SplSlv (Diag,OfDiag,Right,Derv2,NDPts,CrvNum,MxBPts)
C
C This SUBROUTINE solves the diagonally dominant parametric tension
C spline matrix for a given data set using the Gauss-Seidel iteration.
C Convergence is assumed after 20 iterations.
C
      INTEGER  i, j, NDPts(4), CrvNum
C
      REAL Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts),
     $      Derv2(4,MxBPts)
C
C Initialize the second derivative matrix to all zeroes.
C
      DO 10 i=1,NDPts(CrvNum)
         Derv2(CrvNum,i)=0.0
 10   CONTINUE
C
C Calculate the second derivative values using 20 iterations of
C the Gauss-Seidel method.
C
      DO 30 j=1,20
         DO 20 i=2,NDPts(CrvNum)-1
           Derv2(CrvNum,i)=(Right(i)-OfDiag(i)*Derv2(CrvNum,i+1)
     $                              -OfDiag(i-1)*Derv2(CrvNum,i-1))
     $                               /Diag(i)
 20      CONTINUE
 30   CONTINUE
C
      RETURN
      END
C
C=====================================================================
```

```
C
      FUNCTION SplVal (s,w,Derv2,sval,T,n,CrvNum,MxBPts)
C
C This real function finds the w-value (x-value or y-value) corresponding
C to a specified s-value using the parametric tension spline curve
C generated for a particular boundary curve data set.
C
      INTEGER  n, CrvNum
C
      REAL  s(4,MxBPts), w(4,MxBPts), Derv2(4,MxBPts),
     $      sval, T, h, Interim, Temp1, Temp2
C
C
      Temp1=sval-s(CrvNum,n)
      h=s(CrvNum,n+1)-s(CrvNum,n)
      Temp2=s(CrvNum,n+1)-sval
      Interim=Derv2(CrvNum,n)/T**2*SINH(T*Temp2)/SINH(T*h)
     $        +(w(CrvNum,n)-Derv2(CrvNum,n)/T**2)*Temp2/h
      SplVal=Interim+Derv2(CrvNum,n+1)/T**2*SINH(T*Temp1)
     $                      /SINH(T*h)+(w(CrvNum,n+1)
     $                      -Derv2(CrvNum,n+1)/T**2)*Temp1/h
C
      RETURN
      END
C
C=========================================================================
C
      SUBROUTINE PTSpln(x,y,z,s,XDerv2,YDerv2,ZDerv2,Diag,OfDiag,
     $                  Right,NDPts,Tensn,CrvNum,MxBPts)
C
C This SUBROUTINE forms the main routine for the parametric tension
C spline process.
C
      INTEGER  NDPts(4), CrvNum
C
      REAL Diag(MxBPts), OfDiag(MxBPts), Right(MxBPts),
     $     XDerv2(4,MxBPts), YDerv2(4,MxBPts),
     $     ZDerv2(4,MxBPts), Tensn,
     $     x(4,MxBPts), y(4,MxBPts),
     $     z(4,MxBPts), s(4,MxBPts)
C
C
      CALL CalcS(x,y,z,s,NDPts,CrvNum,MxBPts)
      CALL SplMat(Diag,OfDiag,Right,x,s,NDPts,Tensn,CrvNum,MxBPts)
      CALL SplSlv(Diag,OfDiag,Right,XDerv2,NDPts,CrvNum,MxBPts)
      CALL SplMat(Diag,OfDiag,Right,y,s,NDPts,Tensn,CrvNum,MxBPts)
      CALL SplSlv(Diag,OfDiag,Right,YDerv2,NDPts,CrvNum,MxBPts)
      CALL SplMat(Diag,OfDiag,Right,z,s,NDPts,Tensn,CrvNum,MxBPts)
      CALL SplSlv(Diag,OfDiag,Right,ZDerv2,NDPts,CrvNum,MxBPts)
C
      RETURN
      END
C
C=========================================================================
C
      SUBROUTINE FindHs(h1,h2,h3,h4,n,s)
C
C This SUBROUTINE computes the h factors used in Hermite interpolation.
```

```
C
      REAL  h1, h2, h3, h4, n, s
     /      a1, a2, a3, a4, a, b, bbaa, sh, ch, shsn, shsn1
C
      IF(s.NE.0)THEN
         sh=sinh(s)
         ch=cosh(s)
         a2=sh/(2.*sh-s*ch-s)
         a1=1-a2
         a=s*ch-sh
         b=sh-s
         bbaa=b*b-a*a
         a3=-a*sh/bbaa
         a4=b*sh/bbaa
         shsn=sinh(s*n)/sh
         shsn1=sinh(s*(1.-n))/sh
         h1=a2*(shsn1-shsn)+a1*(1.-n)+a2*n
         h2=a2*(shsn-shsn1)+a2*(1.-n)+a1*n
         h3=a3*((1.-n)-shsn1)+a4*(n-shsn)
         h4=a4*(shsn1-(1.-n))+a3*(shsn-n)
      ELSE
         h1= 2*n**3-3*n**2+1
         h2=-2*n**3+3*n**2
         h3= n**3-2*n**2+n
         h4= n**3-n**2
      ENDIF
C
      RETURN
      END
C
C=====================================================================
C
      SUBROUTINE SplInt(n,s,SValue,NDPts,CurCrv,MxBPts)
C
C This SUBROUTINE finds the proper interval in which a point on a specified
C boundary lies.  The interval indicates which initial data points the
C point in question lies between and thus which spline coefficients to
C use.
C
C
      INTEGER  i, n, CurCrv, NDPts(4)
C
      REAL  Temp, SValue, s(4,MxBPts)
C
      n=1
      i=NDPts(CurCrv)
C
 10   IF ((n.EQ.1).AND.(i.GT.1)) THEN
         I=I-1
         Temp=SValue-s(CurCrv,i)
C
         IF (Temp.GT.0.0) THEN
            n=i
         ENDIF
C
         GOTO 10
      ENDIF
C
```

```
      RETURN
      END
C
C=====================================================================
C
      SUBROUTINE FAlNew(AlNew,Alpha,B,Str)
C
C This SUBROUTINE computes the new Alpha value after stretching as
C AlNew.  Alpha is a dummy variable representing either Xi, Eta or Zeta.
C
      INTEGER  Str
C
      REAL  Alpha, Temp1, Temp2, B2, AlNew, B
C
      AlNew=Alpha
      Temp1=(B+1)/(B-1)
C
      IF (Str.EQ.1) THEN
         Temp2=Temp1**(1-Alpha)
         AlNew=((B+1)-(B-1)*Temp2)/(Temp2+1)*1
      ENDIF
C
      IF (Str.EQ.2) THEN
         B2=0
         Temp2=Temp1**((Alpha-B2)/(1-B2))
         AlNew=((B+2*B2)*Temp2-B+2*B2)/((2*B2+1)*(1+Temp2))
      ENDIF
C
      IF (Str.EQ.3) THEN
         B2=0.5
         Temp2=Temp1**((Alpha-B2)/(1-B2))
         AlNew=((B+2*B2)*Temp2-B+2*B2)/((2*B2+1)*(1+Temp2))
      ENDIF
C
      RETURN
      END
C
C=====================================================================
C
      SUBROUTINE EdgPts(X1,X2,X3,X4,Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,AL,BL,
     $                  AAStep,BBStep,x,y,z,s,zx,zy,zz,NDPts,Tensn,
     $                  StrAA,StrBB,BetaAA,BetaBB,MxBPts,MxGSiz)
C
C This SUBROUTINE calculates the grid point locations along the surface
C edges.
C
      INTEGER  ACt, BCt, n1, n2, n3, n4,
     $         AL, BL, StrAA, StrBB, NDPts(4)
C
      REAL  AA, BB, AANew, BBNew, S1, S2, S3, S4, BBStep, AAStep,
     $      S1AAR, S2AAR, S3BBR, S4BBR,
     $      X1(MxGSiz), X2(MxGSiz), X3(MxGSiz), X4(MxGSiz),
     $      Y1(MxGSiz), Y2(MxGSiz), Y3(MxGSiz), Y4(MxGSiz),
     $      Z1(MxGSiz), Z2(MxGSiz), Z3(MxGSiz), Z4(MxGSiz),
     $      x(4,MxBPts),  y(4,MxBPts),  z(4,MxBPts),
     $      s(4,MxBPts), zx(4,MxBPts), zy(4,MxBPts),
     $      zz(4,MxBPts), Tensn(4), BetaAA, BetaBB
C
```

```
      S1AAR=s(1,NDPts(1))
      S2AAR=s(2,NDPts(2))
      S3BBR=s(3,NDPts(3))
      S4BBR=s(4,NDPts(4))
C
C Calculate the grid point locations along boundaries 1 and 2.
C
      AA=0.0
C
      DO 10 ACt=1,AL
         CALL FAlNew(AANew,AA,BetaAA,StrAA)
         S1=AANew*S1AAR
         S2=AANew*S2AAR
         CALL SplInt(n1,s,S1,NDPts,1,MxBPts)
         CALL SplInt(n2,s,S2,NDPts,2,MxBPts)
         X1(ACt)=SplVal(s,x,zx,S1,Tensn(1),n1,1,MxBPts)
         X2(ACt)=SplVal(s,x,zx,S2,Tensn(2),n2,2,MxBPts)
         Y1(ACt)=SplVal(s,y,zy,S1,Tensn(1),n1,1,MxBPts)
         Y2(ACt)=SplVal(s,y,zy,S2,Tensn(2),n2,2,MxBPts)
         Z1(ACt)=SplVal(s,z,zz,S1,Tensn(1),n1,1,MxBPts)
         Z2(ACt)=SplVal(s,z,zz,S2,Tensn(2),n2,2,MxBPts)
         AA=AA+AAStep
 10   CONTINUE
C
C Calculate the grid point locations along boundaries 3 and 4.
C
      BB=0.0
C
      DO 20 BCt=1,BL
         CALL FAlNew(BBNew,BB,BetaBB,StrBB)
         S3=BBNew*S3BBR
         S4=BBNew*S4BBR
         CALL SplInt(n3,s,S3,NDPts,3,MxBPts)
         CALL SplInt(n4,s,S4,NDPts,4,MxBPts)
         X3(BCt)=SplVal(s,x,zx,S3,Tensn(3),n3,3,MxBPts)
         X4(BCt)=SplVal(s,x,zx,S4,Tensn(4),n4,4,MxBPts)
         Y3(BCt)=SplVal(s,y,zy,S3,Tensn(3),n3,3,MxBPts)
         Y4(BCt)=SplVal(s,y,zy,S4,Tensn(4),n4,4,MxBPts)
         Z3(BCt)=SplVal(s,z,zz,S3,Tensn(3),n3,3,MxBPts)
         Z4(BCt)=SplVal(s,z,zz,S4,Tensn(4),n4,4,MxBPts)
         BB=BB+BBStep
 20   CONTINUE
C
      RETURN
      END
C
C======================================================================
C
      SUBROUTINE EdgDer(PX1PAA,PX2PAA,PY1PAA,PY2PAA,PZ1PAA,PZ2PAA,
     $                  PX3PBB,PX4PBB,PY3PBB,PY4PBB,PZ3PBB,PZ4PBB,
     $                  X1,X2,X3,X4,Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,
     $                  AL,BL,MxGSiz)
C
      INTEGER  ACt, BCt, AL, BL
C
      REAL  AAStep, BBStep, PX3PBB(MxGSiz), PX4PBB(MxGSiz),
     $      PY3PBB(MxGSiz), PY4PBB(MxGSiz), PZ3PBB(MxGSiz),
     $      PZ4PBB(MxGSiz), PX1PAA(MxGSiz), PX2PAA(MxGSiz),
```

```
     $      PY1PAA(MxGSiz), PY2PAA(MxGSiz), PZ1PAA(MxGSiz),
     $      PZ2PAA(MxGSiz), X1(MxGSiz), X2(MxGSiz), X3(MxGSiz),
     $      X4(MxGSiz), Y1(MxGSiz), Y2(MxGSiz), Y3(MxGSiz),
     $      Y4(MxGSiz), Z1(MxGSiz), Z2(MxGSiz), Z3(MxGSiz),
     $      Z4(MxGSiz)
C
C Calculate step size in the AA and BB directions.
C
      AAStep=1./(AL-1.)
      BBStep=1./(Bl-1.)
C
C Calculate the derivative values along the constant AA boundaries.
C
      PX1PAA(1)=(X1(2)-X1(1))/AAStep
      PX2PAA(1)=(X2(2)-X2(1))/AAStep
      PY1PAA(1)=(Y1(2)-Y1(1))/AAStep
      PY2PAA(1)=(Y2(2)-Y2(1))/AAStep
      PZ1PAA(1)=(Z1(2)-Z1(1))/AAStep
      PZ2PAA(1)=(Z2(2)-Z2(1))/AAStep
C
      PX1PAA(AL)=(X1(AL)  -X1(AL-1))/AAStep
      PX2PAA(AL)=(X2(AL)  -X2(AL-1))/AAStep
      PY1PAA(AL)=(Y1(AL)  -Y1(AL-1))/AAStep
      PY2PAA(AL)=(Y2(AL)  -Y2(AL-1))/AAStep
      PZ1PAA(AL)=(Z1(AL)  -Z1(AL-1))/AAStep
      PZ2PAA(AL)=(Z2(AL)  -Z2(AL-1))/AAStep
C
      DO 10 ACt=2,AL-1
         PX1PAA(ACt)=  (X1(ACt+1)-X1(ACt-1))/2/AAStep
         PX2PAA(ACt)=  (X2(ACt+1)-X2(ACt-1))/2/AAStep
         PY1PAA(ACt)=  (Y1(ACt+1)-Y1(ACt-1))/2/AAStep
         PY2PAA(ACt)=  (Y2(ACt+1)-Y2(ACt-1))/2/AAStep
         PZ1PAA(ACt)=  (Z1(ACt+1)-Z1(ACt-1))/2/AAStep
         PZ2PAA(ACt)=  (Z2(ACt+1)-Z2(ACt-1))/2/AAStep
 10   CONTINUE
C
C Calculate the derivative values along the constant BB boundaries.
C
      PX3PBB(1)=  (X3(2)-X3(1))/BBStep
      PX4PBB(1)=  (X4(2)-X4(1))/BBStep
      PY3PBB(1)=  (Y3(2)-Y3(1))/BBStep
      PY4PBB(1)=  (Y4(2)-Y4(1))/BBStep
      PZ3PBB(1)=  (Z3(2)-Z3(1))/BBStep
      PZ4PBB(1)=  (Z4(2)-Z4(1))/BBStep
C
      PX3PBB(BL)=(X3(BL)  -X3(BL-1))/BBStep
      PX4PBB(BL)=(X4(BL)  -X4(BL-1))/BBStep
      PY3PBB(BL)=(Y3(BL)  -Y3(BL-1))/BBStep
      PY4PBB(BL)=(Y4(BL)  -Y4(BL-1))/BBStep
      PZ3PBB(BL)=(Z3(BL)  -Z3(BL-1))/BBStep
      PZ4PBB(BL)=(Z4(BL)  -Z4(BL-1))/BBStep
C
      DO 20 BCt=2,BL-1
         PX3PBB(BCt)=  (X3(BCt+1)-X3(BCt-1))/2/BBStep
         PX4PBB(BCt)=  (X4(BCt+1)-X4(BCt-1))/2/BBStep
         PY3PBB(BCt)=  (Y3(BCt+1)-Y3(BCt-1))/2/BBStep
         PY4PBB(BCt)=  (Y4(BCt+1)-Y4(BCt-1))/2/BBStep
         PZ3PBB(BCt)=  (Z3(BCt+1)-Z3(BCt-1))/2/BBStep
```

```
         PZ4PBB(BCt)=  (Z4(BCt+1)-Z4(BCt-1))/2/BBStep
 20   CONTINUE
C
      RETURN
      END
C
C======================================================================
C
      SUBROUTINE TwoBnd(XS,YS,ZS,SrfNum,AL,BL,SigmaBB,k1,k2,
     $                 StrB,h1,h2,h3,h4,X1,X2,X3,X4,
     $                 Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,PX1PBB,PX2PBB,
     $                 PY1PBB,PY2PBB,PZ1PBB,PZ2PBB,PX1PAA,PX2PAA,
     $                 PY1PAA,PY2PAA,PZ1PAA,PZ2PAA,PX3PBB,PX4PBB,
     $                 PY3PBB,PY4PBB,PZ3PBB,PZ4PBB,
     $                 MxBCvs,MxGSiz,MxSrfs)
C
C This SUBROUTINE calculates the interior grid point locations between
C two specified boundaries (1 and 2) by using transfinite Hermite
C interpolation.
C
      INTEGER  ACt, BCt, AL, BL, StrAA, StrBB, SrfNum, Edg1, Edg2,
     $         Edg3, Edg4
C
      REAL  AA, BB, AANew, BBNew, LL1, LL2,
     $      Box1i, Box1j, Box1k, Box2i, Box2j, Box2k,
     $      Temp1i, Temp1j, Temp1k, Temp2i, Temp2j, Temp2k,
     $      k1(MxSrfs,MxGSiz), k2(MxSrfs,MxGSiz),
     $      BetaAA, BetaBB, BBStep, AAStep,
     $      h1(MxGSiz), h2(MxGSiz), h3(MxGSiz), h4(MxGSiz),
     $      X1(MxGSiz), X2(MxGSiz), X3(MxGSiz), X4(MxGSiz),
     $      Y1(MxGSiz), Y2(MxGSiz), Y3(MxGSiz), Y4(MxGSiz),
     $      Z1(MxGSiz), Z2(MxGSiz), Z3(MxGSiz), Z4(MxGSiz)
      REAL  PX1PBB(MxGSiz), PX2PBB(MxGSiz),
     $      PY1PBB(MxGSiz), PY2PBB(MxGSiz),
     $      PZ1PBB(MxGSiz), PZ2PBB(MxGSiz),
     $      PX1PAA(MxGSiz), PX2PAA(MxGSiz),
     $      PY1PAA(MxGSiz), PY2PAA(MxGSiz),
     $      PZ1PAA(MxGSiz), PZ2PAA(MxGSiz),
     $      PX3PBB(MxGSiz), PX4PBB(MxGSiz),
     $      PY3PBB(MxGSiz), PY4PBB(MxGSiz),
     $      PZ3PBB(MxGSiz), PZ4PBB(MxGSiz),
     $      XS(MxSrfs,MxGSiz,MxGSiz),
     $      YS(MxSrfs,MxGSiz,MxGSiz),
     $      ZS(MxSrfs,MxGSiz,MxGSiz),
     $      StrB(MxGSiz,MxBCvs)
C
C Calculate the step size in the AA and BB directions.
C
      AAStep=1./(AL-1.)
      BBStep=1./(BL-1.)
C
C
C Calculate the edge numbers for the surface 'SrfNum'.
C
      Edg1=(SrfNum-1)*4 + 1
      Edg2=(SrfNum-1)*4 + 2
      Edg3=(SrfNum-1)*4 + 3
      Edg4=(SrfNum-1)*4 + 4
```

```
C
C Calculate the derivative values for grid line orthogonality.
C
      AA=0.0
C
      DO 20 ACt=1,AL
         Box1i=      AA*( PY1PAA(AL)*PZ4PBB(1)
     $                   -PZ1PAA(AL)*PY4PBB(1))
     $           +(1-AA)*( PY1PAA(1)  *PZ3PBB(1)
     $                   -PZ1PAA(1)  *PY3PBB(1))
         Box1j=      AA*( PX1PAA(AL)*PZ4PBB(1)
     $                   -PZ1PAA(AL)*PX4PBB(1))
     $           +(1-AA)*( PX1PAA(1)  *PZ3PBB(1)
     $                   -PZ1PAA(1)  *PX3PBB(1))
         Box1k=      AA*( PX1PAA(AL)*PY4PBB(1)
     $                   -PY1PAA(AL)*PX4PBB(1))
     $           +(1-AA)*( PX1PAA(1)  *PY3PBB(1)
     $                   -PY1PAA(1)  *PX3PBB(1))
         Box2i=      AA*( PY2PAA(AL)*PZ4PBB(BL)
     $                   -PZ2PAA(AL)*PY4PBB(BL))
     $           +(1-AA)*( PY2PAA(1)  *PZ3PBB(BL)
     $                   -PZ2PAA(1)  *PY3PBB(BL))
         Box2j=      AA*( PX2PAA(AL)*PZ4PBB(BL)
     $                   -PZ2PAA(AL)*PX4PBB(BL))
     $           +(1-AA)*( PX2PAA(1)  *PZ3PBB(BL)
     $                   -PZ2PAA(1)  *PX3PBB(BL))
         Box2k=      AA*( PX2PAA(AL)*PY4PBB(BL)
     $                   -PY2PAA(AL)*PX4PBB(BL))
     $           +(1-AA)*( PX2PAA(1)  *PY3PBB(BL)
     $                   -PY2PAA(1)  *PX3PBB(BL))
         Temp1i=PY1PAA(ACt)*Box1k+PZ1PAA(ACt)*Box1j
         Temp1j=PX1PAA(ACt)*Box1k-PZ1PAA(ACt)*Box1i
         Temp1k=PX1PAA(ACt)*Box1j+PY1PAA(ACt)*Box1i
         Temp2i=PY2PAA(ACt)*Box2k+PZ2PAA(ACt)*Box2j
         Temp2j=PX2PAA(ACt)*Box2k-PZ2PAA(ACt)*Box2i
         Temp2k=PX2PAA(ACt)*Box2j+PY2PAA(ACt)*Box2i
         LL1=(Temp1i**2+Temp1j**2+Temp1k**2)**0.5
         LL2=(Temp2i**2+Temp2j**2+Temp2k**2)**0.5
C
         PX1PBB(ACt)=-k1(SrfNum,ACt)*Temp1i/LL1
         PX2PBB(ACt)=-k2(SrfNum,ACt)*Temp2i/LL2
         PY1PBB(ACt)= k1(SrfNum,ACt)*Temp1j/LL1
         PY2PBB(ACt)= k2(SrfNum,ACt)*Temp2j/LL2
         PZ1PBB(ACt)= k1(SrfNum,ACt)*Temp1k/LL1
         PZ2PBB(ACt)= k2(SrfNum,ACt)*Temp2k/LL2
C
         AA=AA+AAStep
 20   CONTINUE
C
C Calculate the interior grid point locations.
C
      DO 40 ACt=1,AL
         DO 30 BCt=1,BL
            AA=(ACt-1.)/(AL-1.)
            BBNew=StrB(BCt,Edg3)*(1.-AA)+StrB(BCt,Edg4)*AA
            CALL FindHs(h1(BCt),h2(BCt),h3(BCt),h4(BCt),BBNew,SigmaBB)
            XS(SrfNum,ACt,BCt)= h1(BCt)*X1(ACt)+h2(BCt)*X2(ACt)
     $                         +h3(BCt)*PX1PBB(ACt)+h4(BCt)*PX2PBB(ACt)
```

```
            YS(SrfNum,ACt,BCt)= h1(BCt)*Y1(ACt)+h2(BCt)*Y2(ACt)
     $                        +h3(BCt)*PY1PBB(ACt)+h4(BCt)*PY2PBB(ACt)
            ZS(SrfNum,ACt,BCt)= h1(BCt)*Z1(ACt)+h2(BCt)*Z2(ACt)
     $                        +h3(BCt)*PZ1PBB(ACt)+h4(BCt)*PZ2PBB(ACt)
 30       CONTINUE
 40     CONTINUE
C
      RETURN
      END
C
C=====================================================================
C
      SUBROUTINE ForBnd(XS,YS,ZS,SrfNum,AL,BL,SigmaAA,SigmaBB,k3,k4,
     $                  StrB,h1,h2,h3,h4,h5,h6,h7,h8,
     $                  X1,X2,X3,X4,Y1,Y2,Y3,Y4,Z1,Z2,Z3,Z4,
     $                  PX1PBB,PX2PBB,PY1PBB,PY2PBB,PZ1PBB,PZ2PBB,
     $                  PX1PAA,PX2PAA,PY1PAA,PY2PAA,PZ1PAA,PZ2PAA,
     $                  PX3PBB,PX4PBB,PY3PBB,PY4PBB,PZ3PBB,PZ4PBB,
     $                  PX3PAA,PX4PAA,PY3PAA,PY4PAA,PZ3PAA,PZ4PAA,
     $                  MxBCvs,MxGSiz,MxSrfs)
C
C This SUBROUTINE adjusts the grid so that the other two boundaries
C (3 and 4) of the surface are mapped correctly using transfinite Hermite
C interpolation.
C
      INTEGER  ACt, BCt, AL, BL, StrAA, StrBB, i, j, SrfNum,
     $         Edg1, Edg2, Edg3, Edg4
C
      REAL  AA, BB, AANew, BBNew, LL3, LL4,
     $      Box3i, Box3j, Box3k, Box4i, Box4j, Box4k,
     $      Temp3i, Temp3j, Temp3k, Temp4i, Temp4j, Temp4k,
     $      P2Y00, P2Y01, P2Y10, P2Y11, P2X00, P2X01, P2X10, P2X11,
     $      P2Z00, P2Z01, P2Z10, P2Z11,
     $      k3(MxSrfs,MxGSiz), k4(MxSrfs,MxGSiz),
     $      BetaAA, BetaBB, BBStep, AAStep,
     $      h1(MxGSiz), h2(MxGSiz), h3(MxGSiz), h4(MxGSiz),
     $      h5(MxGSiz), h6(MxGSiz), h7(MxGSiz), h8(MxGSiz),
     $      X1(MxGSiz), X2(MxGSiz), X3(MxGSiz), X4(MxGSiz),
     $      Y1(MxGSiz), Y2(MxGSiz), Y3(MxGSiz), Y4(MxGSiz),
     $      Z1(MxGSiz), Z2(MxGSiz), Z3(MxGSiz), Z4(MxGSiz)
      REAL  PX1PBB(MxGSiz), PX2PBB(MxGSiz),
     $      PY1PBB(MxGSiz), PY2PBB(MxGSiz),
     $      PZ1PBB(MxGSiz), PZ2PBB(MxGSiz),
     $      PX1PAA(MxGSiz), PX2PAA(MxGSiz),
     $      PY1PAA(MxGSiz), PY2PAA(MxGSiz),
     $      PZ1PAA(MxGSiz), PZ2PAA(MxGSiz),
     $      PX3PBB(MxGSiz), PX4PBB(MxGSiz),
     $      PY3PBB(MxGSiz), PY4PBB(MxGSiz),
     $      PZ3PBB(MxGSiz), PZ4PBB(MxGSiz),
     $      PX3PAA(MxGSiz), PX4PAA(MxGSiz),
     $      PY3PAA(MxGSiz), PY4PAA(MxGSiz),
     $      PZ3PAA(MxGSiz), PZ4PAA(MxGSiz),
     $      XS(MxSrfs,MxGSiz,MxGSiz),
     $      YS(MxSrfs,MxGSiz,MxGSiz),
     $      ZS(MxSrfs,MxGSiz,MxGSiz),
     $      StrB(MxGSiz,MxBCvs)
C
C Calculate the step size for directions AA and BB.
```

```
C
      AAStep=1./(AL-1.)
      BBStep=1./(BL-1.)
C
C Calculate edge numbers for the surface 'SrfNum'
C
      Edg1=(SrfNum-1)*4 + 1
      Edg2=(SrfNum-1)*4 + 2
      Edg3=(SrfNum-1)*4 + 3
      Edg4=(SrfNum-1)*4 + 4
C
C
C Calculate the derivative values for grid line orthogonality.
C
      BB=0.0
C
      DO 20 BCt=1,BL
         Box3i=      BB*( PY2PAA(1)*PZ3PBB(BL)
     $                   -PZ2PAA(1)*PY3PBB(BL))
     $          +(1-BB)*( PY1PAA(1)*PZ3PBB(1)
     $                   -PZ1PAA(1)*PY3PBB(1))
         Box3j=      BB*( PX2PAA(1)*PZ3PBB(BL)
     $                   -PZ2PAA(1)*PX3PBB(BL))
     $          +(1-BB)*( PX1PAA(1)*PZ3PBB(1)
     $                   -PZ1PAA(1)*PX3PBB(1))
         Box3k=      BB*( PX2PAA(1)*PY3PBB(BL)
     $                   -PY2PAA(1)*PX3PBB(BL))
     $          +(1-BB)*( PX1PAA(1)*PY3PBB(1)
     $                   -PY1PAA(1)*PX3PBB(1))
         Box4i=      BB*( PY2PAA(1)*PZ4PBB(BL)
     $                   -PZ2PAA(1)*PY4PBB(BL))
     $          +(1-BB)*( PY1PAA(1)*PZ4PBB(1)
     $                   -PZ1PAA(1)*PY4PBB(1))
         Box4j=      BB*( PX2PAA(1)*PZ4PBB(BL)
     $                   -PZ2PAA(1)*PX4PBB(BL))
     $          +(1-BB)*( PX1PAA(1)*PZ4PBB(1)
     $                   -PZ1PAA(1)*PX4PBB(1))
         Box4k=      BB*( PX2PAA(1)*PY4PBB(BL)
     $                   -PY2PAA(1)*PX4PBB(BL))
     $          +(1-BB)*( PX1PAA(1)*PY4PBB(1)
     $                   -PY1PAA(1)*PX4PBB(1))
         Temp3i=PZ3PBB(BCt)*Box3j+PY3PBB(BCt)*Box3k
         Temp3j=PZ3PBB(BCt)*Box3i-PX3PBB(BCt)*Box3k
         Temp3k=PY3PBB(BCt)*Box3i+PX3PBB(BCt)*Box3j
         Temp4i=PZ4PBB(BCt)*Box4j+PY4PBB(BCt)*Box4k
         Temp4j=PZ4PBB(BCt)*Box4i-PX4PBB(BCt)*Box4k
         Temp4k=PY4PBB(BCt)*Box4i+PX4PBB(BCt)*Box4j
         LL3=(Temp3i**2+Temp3j**2+Temp3k**2)**0.5
         LL4=(Temp4i**2+Temp4j**2+Temp4k**2)**0.5
C
         PX3PAA(BCt)= k3(SrfNum,BCt)*Temp3i/LL3
         PX4PAA(BCt)= k4(SrfNum,BCt)*Temp4i/LL4
         PY3PAA(BCt)= k3(SrfNum,BCt)*Temp3j/LL3
         PY4PAA(BCt)= k4(SrfNum,BCt)*Temp4j/LL4
         PZ3PAA(BCt)=-k3(SrfNum,BCt)*Temp3k/LL3
         PZ4PAA(BCt)=-k4(SrfNum,BCt)*Temp4k/LL4
C
          BB=BB+BBStep
```

```
 20   CONTINUE
C
C Set the cross-derivative terms equal to zero.
C
      P2X00=0.0
      P2X10=0.0
      P2X01=0.0
      P2X11=0.0
      P2Y00=0.0
      P2Y10=0.0
      P2Y01=0.0
      P2Y11=0.0
      P2Z00=0.0
      P2Z10=0.0
      P2Z01=0.0
      P2Z11=0.0
C
C Calculate the grid point locations everywhere.
C
      DO 40 i=1,AL
         DO 30 j=1,BL
            AA=(i-1.)/(AL-1.)
            BB=(j-1.)/(BL-1.)
            AANew=StrB(i,Edg1)*(1.-BB)+StrB(i,Edg2)*BB
            BBNew=StrB(j,Edg3)*(1.-AA)+StrB(j,Edg4)*AA
            CALL FindHs(h1(j),h2(j),h3(j),h4(j),BBNew,SigmaBB)
            CALL FindHs(h5(i),h6(i),h7(i),h8(i),AANew,SigmaAA)
            XS(SrfNum,i,j)=XS(SrfNum,i,j)
     $              +(X3(j)-h1(j)*X1(1)
     $                     -h2(j)*X2(1)
     $                     -h3(j)*PX1PBB(1)
     $                     -h4(j)*PX2PBB(1))*h5(i)
     $              +(X4(j)-h1(j)*X1(AL)
     $                     -h2(j)*X2(AL)
     $                     -h3(j)*PX1PBB(AL)
     $                     -h4(j)*PX2PBB(AL))*h6(i)
     $              +(PX3PAA(j)-( h1(j)*PX3PAA(1)
     $                           +h2(j)*PX3PAA(BL)
     $                           +h3(j)*P2X00+h4(j)*P2X01))*h7(i)
     $              +(PX4PAA(j)-( h1(j)*PX4PAA(1)
     $                           +h2(j)*PX4PAA(BL)
     $                           +h3(j)*P2X10+h4(j)*P2X11))*h8(i)
            YS(SrfNum,i,j)=YS(SrfNum,i,j)
     $              +(Y3(j)-h1(j)*Y1(1)
     $                     -h2(j)*Y2(1)
     $                     -h3(j)*PY1PBB(1)
     $                     -h4(j)*PY2PBB(1))*h5(i)
     $              +(Y4(j)-h1(j)*Y1(AL)
     $                     -h2(j)*Y2(AL)
     $                     -h3(j)*PY1PBB(AL)
     $                     -h4(j)*PY2PBB(AL))*h6(i)
     $              +(PY3PAA(j)-( h1(j)*PY3PAA(1)
     $                           +h2(j)*PY3PAA(BL)
     $                           +h3(j)*P2Y00+h4(j)*P2Y01))*h7(i)
     $              +(PY4PAA(j)-( h1(j)*PY4PAA(1)
     $                           +h2(j)*PY4PAA(BL)
     $                           +h3(j)*P2Y10+h4(j)*P2Y11))*h8(i)
            ZS(SrfNum,i,j)=ZS(SrfNum,i,j)
```

```
     $                 +(Z3(j)-h1(j)*Z1(1)
     $                        -h2(j)*Z2(1)
     $                        -h3(j)*PZ1PBB(1)
     $                        -h4(j)*PZ2PBB(1))*h5(i)
     $                 +(Z4(j)-h1(j)*Z1(AL)
     $                        -h2(j)*Z2(AL)
     $                        -h3(j)*PZ1PBB(AL)
     $                        -h4(j)*PZ2PBB(AL))*h6(i)
     $                 +(PZ3PAA(j)-( h1(j)*PZ3PAA(1)
     $                              +h2(j)*PZ3PAA(BL)
     $                              +h3(j)*P2Z00+h4(j)*P2Z01))*h7(i)
     $                 +(PZ4PAA(j)-( h1(j)*PZ4PAA(1)
     $                              +h2(j)*PZ4PAA(BL)
     $                              +h3(j)*P2Z10+h4(j)*P2Z11))*h8(i)
 30         CONTINUE
 40     CONTINUE
C
        RETURN
        END
C
C=======================================================================
        SUBROUTINE XiEtFl (XPnt,YPnt,ZPnt,II,JJ,KK,J1,J2,NKCsps,KCusp,
     $                     NoIts,MxGSiz)
C
C This SUBROUTINE smooths 3D, constant Zeta grid planes which have been
C disturbed by a constant Zeta cusp in a Xi-Zeta boundary surface.  The
C process produces smoother grid lines in the Zeta direction.
C
        INTEGER  kc, i, j, k, l, II, JJ, KK, NICsps, NoIts,
     $           KCusp(MxGSiz), J1, J2
C
C       REAL  XPnt(MxGSiz,MxGSiz,MxGSiz), YPnt(MxGSiz,MxGSiz,MxGSiz),
C      $      ZPnt(MxGSiz,MxGSiz,MxGSiz)
        REAL XPnt(II,JJ,KK),YPnt(II,JJ,KK),ZPnt(II,JJ,KK)
C
C
        DO 30 k=1,NKCsps
          kc=KCusp(k)
          DO 20 i=2,II-1
            DO 10 j=J1+1,J2-1
              DO 5 l=1,NoIts
                XPnt(i,j,kc)=0.5*(XPnt(i,j,kc+1)-2*XPnt(i,j,kc)
     $                           +XPnt(i,j,kc-1))+ XPnt(i,j,kc)
                YPnt(i,j,kc)=0.5*(YPnt(i,j,kc+1)-2*YPnt(i,j,kc)
     $                           +YPnt(i,j,kc-1))+ YPnt(i,j,kc)
                ZPnt(i,j,kc)=0.5*(ZPnt(i,j,kc+1)-2*ZPnt(i,j,kc)
     $                           +ZPnt(i,j,kc-1))+ ZPnt(i,j,kc)
                XPnt(i,j,kc-1)=0.25*(XPnt(i,j,kc)-2* XPnt(i,j,kc-1)
     $                           +XPnt(i,j,kc-2))+XPnt(i,j,kc-1)
                YPnt(i,j,kc-1)=0.25*(YPnt(i,j,kc)-2* YPnt(i,j,kc-1)
     $                           +YPnt(i,j,kc-2))+YPnt(i,j,kc-1)
                ZPnt(i,j,kc-1)=0.25*(ZPnt(i,j,kc)-2* ZPnt(i,j,kc-1)
     $                           +ZPnt(i,j,kc-2))+ZPnt(i,j,kc-1)
                XPnt(i,j,kc+1)=0.25*(XPnt(i,j,kc+2)-2*XPnt(i,j,kc+1)
     $                           +XPnt(i,j,kc))+   XPnt(i,j,kc+1)
                YPnt(i,j,kc+1)=0.25*(YPnt(i,j,kc+2)-2*YPnt(i,j,kc+1)
     $                           +YPnt(i,j,kc))+   YPnt(i,j,kc+1)
                ZPnt(i,j,kc+1)=0.25*(ZPnt(i,j,kc+2)-2*ZPnt(i,j,kc+1)
```

```
     $                                    +ZPnt(i,j,kc))+    ZPnt(i,j,kc+1)
 5             CONTINUE
 10          CONTINUE
 20        CONTINUE
 30    CONTINUE
C
      RETURN
      END
C
C=====================================================================#
C
      SUBROUTINE RdGrPIn(NGPts,XB,YB,ZB,CrvNum,MxGSiz,InNum)
C
      INTEGER NGPts, CrvNum
C
      REAL    XB(MxGSiz,4), YB(MxGSiz,4), ZB(MxGSiz,4)
C
C This subroutine reads in the coordinates of the grid points
C on one edge.
C
      DO 10 i=1,NGPts
         READ(InNum,*)XB(i,CrvNum),YB(i,CrvNum),ZB(i,CrvNum)
 10   CONTINUE
C
      RETURN
      END
C
C=====================================================================#
C
      SUBROUTINE CalStl(NGPts,XB,YB,ZB,CrvNum,EdgNum,
     $                         StrB,MxBCvs,MxGSiz)
C
C This subroutine calculates the distribution function that corresponds
C to the given distribution of grid points along the edge 'EdgNum'.
C
      INTEGER NGPts, CrvNum, EdgNum, i
C
      REAL    XB(MxGSiz,4), YB(MxGSiz,4), ZB(MxGSiz,4),
     $        StrB(MxGSiz,MxBCvs)
C
      StrB(1,EdgNum)=0.
C
      DO 10 i=2,NGPts
         StrB(i,EdgNum)=StrB(i-1,EdgNum) +
     $                   SQRT((XB(i,CrvNum)-XB(i-1,CrvNum))**2 +
     $                        (YB(i,CrvNum)-YB(i-1,CrvNum))**2 +
     $                        (ZB(i,CrvNum)-ZB(i-1,CrvNum))**2)
 10   CONTINUE
C
      SMax=StrB(NGPts,EdgNum)
      DO 20 i=2,NGPts
         StrB(i,EdgNum)=StrB(i,EdgNum)/SMax
 20   CONTINUE
C
      RETURN
      END
C
C=====================================================================#
```

```
C
      SUBROUTINE CalSt2(EdgNum,NGPts,StrTp,Beta1,Beta2,
     $                       StrB,MxBCvs,MxGSiz)
C
C This subroutine calculates the distribution function based on the
C stretching parameters 'StrTp' and 'Beta'
C
      INTEGER NGPts, StrTp, EdgNum, i
C
      REAL    StrB(MxGSiz,MxBCvs), Beta1, Beta2, A, B, DZ
C
      StrB(1,EdgNum)=0.
      IF(StrTp.LE.3)THEN
         DO 10 i=1,NGPts-1
            Alpha=(i-1.)/(NGPts-1.)
            CALL FAlNew(AlNew,Alpha,Beta1,StrTp)
            StrB(i,EdgNum)=AlNew
10       CONTINUE
      ELSEIF(StrTp.EQ.4)THEN
         CALL Str4Prm(Beta1,Beta2,A,B,DZ)
         DO 20 i=2,NGPts-1
            Alpha=(i-1.)/(NGPts-1.)
            CALL Str4(AlNew,Alpha,A,B,DZ)
            StrB(i,EdgNum)=AlNew
20       CONTINUE
      ENDIF
      StrB(NGPts,EdgNum)=1.
C
      RETURN
      END
C
C=====================================================================#
C
      SUBROUTINE EdgGPts(CrvNum,EdgNum,NGPts,XB,YB,ZB,StrB,
     $                      x,y,z,s,zx,zy,zz,NDPts,Tensn,
     $                      MxBCvs,MxBPts,MxGSiz)
C
C This subroutine calculates the grid point location along an edge
C based on a spline curve fitted through specified nodal points and a
C given distribution function.
C
      INTEGER CrvNum, EdgNum, NGPts, NDPts(4), i, n
C
      REAL    XB(MxGSiz,4), YB(MxGSiz,4), ZB(MxGSiz,4),
     $        StrB(MxGSiz,MxBCvs), x(4,MxBPts), y(4,MxBPts),
     $        z(4,MxBPts), zx(4,MxBPts), zy(4,MxBPts),
     $        zz(4,MxBPts), s(4,MxBPts), Tensn
C
      SRa=S(CrvNum,NDPts(CrvNum))
C
      DO 10 i=1,NGPts
         SB=SRa*StrB(i,EdgNum)
         CALL SplInt(n,s,SB,NDPts,CrvNum,MxBPts)
         XB(i,CrvNum)=SplVal(s,x,zx,SB,Tensn,n,CrvNum,MxBPts)
         YB(i,CrvNum)=SplVal(s,y,zy,SB,Tensn,n,CrvNum,MxBPts)
         ZB(i,CrvNum)=SplVal(s,z,zz,SB,Tensn,n,CrvNum,MxBPts)
10    CONTINUE
C
```

```
      RETURN
      END
C
C=====================================================================
C
      SUBROUTINE Str4Prm(S0,S1,A,B,DZ)
C
      REAL S0, S1, A, B, DZ, Y, PI
C
C  This subroutine calculates the parameters A, B, and DZ for the two-
C  sided Vinokur stretching function.
C
      PI=ACOS(-1.)
C
      A=SQRT(S0/S1)
      B=SQRT(S0*S1)
C
      IF(B.GT.1.001)THEN
         IF(B.LE.2.7829681)THEN
            Y=B-1
            DZ=SQRT(6.*Y)*(1.-0.15*Y+0.057321429*(Y**2)
     $                    -0.024907295*(Y**3)+0.0077424461*(Y**4)
     $                    -0.0010794123*(Y**5))
         ELSEIF(B.GT.2.7829681)THEN
            V=LOG(B)
            W=1./B - 0.028527431
            DZ=V+(1.+1./V)*LOG(2.*V)-0.02041793+0.24902722*W
     $         +1.9496443*(W**2)-2.6294547*(W**3)+8.56795911*(W**4)
         ENDIF
      ELSEIF(B.LT.0.999)THEN
         IF(B.LE.0.26938972)THEN
            DZ=PI*(1.-B+B**2-(1.+(PI**2)/6.)*(B**3)+6.794732*(B**4)
     $         -13.205501*(B**5)+11.726095*(B**6))
         ELSE
            Y=B-1
            DZ=SQRT(6.*Y)*(1.+0.15*Y+0.057321429*(Y**2)
     $                    +0.048774238*(Y**3)-0.053337753*(Y**4)
     $                    +0.075845134*(Y**5))
         ENDIF
      ENDIF
C
C
      RETURN
      END
C
C=====================================================================
C
      SUBROUTINE Str4(AlNew,Alpha,A,B,DZ)
C
      REAL AlNew, Alpha, A, B, DZ, U, T
C
C  This subroutine calculates the value of the two-sided Vinokur
C  stretching function based on the value of the parameters A, B,
C  and DZ, and on the value of the "computational" coordinate Alpha.
C
C
      IF(B.GT.1.001)THEN
        U=0.5+TANH(DZ*(Alpha-0.5))/(2.*TANH(DZ/2.))
```

```
      ELSEIF(B.LT.0.999)THEN
        U=0.5+TAN(DZ*(Alpha-0.5))/(2.*TAN(DZ/2.))
      ELSE
        U=Alpha*(1.+2.*(B-1)*(Alpha-0.5)*(1-Alpha))
      ENDIF
      T=U/(A+(1.-A)*U)
      AlNew=T
C
C
      RETURN
      END
C
C========================================================================
C
      SUBROUTINE KFctrs(ZoneNo,kS,k1,k2,k3,k4,kXi1,kXi2,kEta1,kEta2,
     $                  kZeta1,kZeta2,MxSrfs,MxGSiz)
C
C  This subroutine is used to set the k-factors that are to be used.
C  The value of the k-factors is first set equal to the user specified
C  values of kXi1, kXi2, kEta1, kEta2, kZeta1 and kZeta2.  After that
C  the user can modify the k-factors for individual grid lines as
C  desired.
C
      INTEGER ZoneNo
C
      REAL kS(MxSrfs,MxGSiz,MxGSiz), k1(MxSrfs,MxGsiz),
     $     k2(MxSrfs,MxGSiz), k3(MxSrfs,MxGSiz), k4(MxSrfs,MxGSiz),
     $     kXi1, kXi2, kEta1, kEta2, kZeta1, kZeta2
C
C
C   Set the starting values of the k-factors:
C
C     first, k-factors used in generating interior grid points
C
      DO 100 i1=1,MxGSiz
      DO 100 i2=1,MxGSiz
         kS(1,i1,i2)=kEta1
         kS(2,i1,i2)=kEta2
         kS(3,i1,i2)=kZeta1
         kS(4,i1,i2)=kZeta2
100   CONTINUE
C
C     then, k-factors used to generate boundary surfaces.
C
      DO 200 i1=1,MxGSiz
         k1(1,i1)=kXi1
         k2(1,i1)=kXi2
         k3(1,i1)=kZeta1
         k4(1,i1)=kZeta2
         k1(2,i1)=kXi1
         k2(2,i1)=kXi2
         k3(2,i1)=kZeta1
         k4(2,i1)=kZeta2
         k1(3,i1)=kEta1
         k2(3,i1)=kEta2
         k3(3,i1)=kXi1
         k4(3,i1)=kXi2
         k1(4,i1)=kEta1
```

```
         k2(4,i1)=kEta2
         k3(4,i1)=kXi1
         k4(4,i1)=kXi2
200   CONTINUE
C
C
C  Here, the user can make any desired modification of the K-
C  factors to improve the grid that he/she is generating.  This part
C  of the subroutine will be case dependent.
C
C
C
C
      RETURN
      END
```

REFERENCES

1. Shih, T.I.-P., et al.: GRID2D/3D - A Computer Program for Generating Grid Systems in Complex-Shaped Two- and Three-Dimensional Spatial Domains: Part 1 - Theory and Method. NASA TM-102453, 1990.

2. Bailey, R.T., et al.: GRID2D/3D - A Computer Program for Generating Grid Systems in Complex-Shaped Two- and Three-Dimensional Spatial Domains: Part 2 - User's Manual and Program Listing. NASA TM-102454, 1990.

3. Vinokur, M.: On One-Dimensional Stretching Functions for Finite-Difference Calculations. J. Comput. Phys., vol. 50, May 1983, pp. 215-234. (Also, NASA CR-3313.)

4. Thompson, J.F.; Warsi, Z.U.A.; and Mastin, C.W.: Numerical Grid Generation. Elsevier Science Publishing Co., 1985.

TABLE 3.1 — Guide To Grid Control Parameters

Parameter	Range	Trial value	Function
n	2 (two boundary method) 4 (four boundary method)		Controls method used to generate the grid
σ_ξ σ_η σ_ζ	$0 \le \sigma \le \infty$	0 to 10	Controls the shape (curvature) of grid lines
$K_{\xi 1}$ $K_{\xi 2}$ $K_{\eta 1}$ $K_{\eta 2}$ $K_{\zeta 1}$ $K_{\zeta 2}$ (K-factors)	$0 \le K \le \infty$	10^{-3}	Control orthogonality at boundaries and curvature of grid lines
Type-i	1 (grid points) 2 (node points and stretching function)		Determines whether edge is defined using grid points or node points for splining
StretchType i	0 (no clustering) 1 (clustering near lower boundary) 2 (clustering near upper boundary) 3 (symmetric clustering near both boundaries) 4 (asymmetric clustering near both boundaries)		Controls the distribution of grid points along Edge i
Beta1	For StretchType =1,2 or 3: $1 < \text{Beta1} < \infty$	1.1	Controls amount of clustering near boundaries (clustering increases as Beta1 $\rightarrow$ 1)
Beta1 Beta2	For StretchType = 4: $0 < \text{Beta} < \infty$	2.5	Control amount of clustering near boundaries (clustering increases as Beta $\rightarrow \infty$)

TABLE 3.2 — **Listing of Input File For Generation of Grid System For Zone 18 of Radial Turbine Coolant Passage**

```
          4    Technique
         10     IL
         15     JL
         49     KL
     10.000000      SigmaXi
     20.000000      SigmaEta
     10.000000      SigmaZeta
  0.000000E+00      kXI1
  0.000000E+00      kXI2
  2.500000E-03      kETA1
  2.500000E-03      kETA2
  5.000000E-03      kZETA1
  5.000000E-03      kZETA2
 1      Type - EDGE NO:  1 -----------------
   .11645      .01677      .05842      -- 1
   .11628      .01675      .05843      -- 2
   .11605      .01672      .05844      -- 3
   .11572      .01668      .05845      -- 4
   .11532      .01660      .05843      -- 5
   .11491      .01664      .05853      -- 6
   .11467      .01690      .05878      -- 7
   .11459      .01717      .05902      -- 8
   .11438      .01715      .05904      -- 9
   .11392      .01711      .05907      --10
   .11321      .01704      .05911      --11
   .11250      .01698      .05915      --12
   .11204      .01694      .05918      --13
   .11183      .01692      .05920      --14
   .11181      .01662      .05894      --15
   .11170      .01633      .05870      --16
   .11148      .01608      .05851      --17
   .11114      .01592      .05842      --18
   .11078      .01598      .05853      --19
   .11056      .01620      .05877      --20
   .11047      .01648      .05902      --21
   .11026      .01646      .05904      --22
   .10980      .01642      .05907      --23
   .10909      .01635      .05911      --24
   .10839      .01627      .05915      --25
   .10793      .01623      .05918      --26
   .10772      .01621      .05920      --27
   .10770      .01593      .05894      --28
   .10759      .01567      .05869      --29
   .10736      .01543      .05850      --30
   .10702      .01529      .05842      --31
   .10667      .01535      .05853      --32
   .10645      .01555      .05876      --33
   .10636      .01581      .05902      --34
   .10615      .01579      .05904      --35
   .10569      .01575      .05907      --36
   .10498      .01568      .05911      --37
   .10427      .01560      .05915      --38
```

TABLE 3.2 (continued)

.10381	.01556	.05918	--39
.10360	.01554	.05920	--40
.10359	.01531	.05898	--41
.10350	.01504	.05875	--42
.10331	.01479	.05854	--43
.10300	.01460	.05842	--44
.10268	.01450	.05838	--45
.10239	.01444	.05838	--46
.10214	.01442	.05839	--47
.10195	.01440	.05841	--48
.10181	.01439	.05842	--49
1 Type - EDGE NO: 2 -----------------			
.11596	.01985	.05842	-- 1
.11583	.01983	.05842	-- 2
.11564	.01981	.05843	-- 3
.11538	.01977	.05843	-- 4
.11507	.01975	.05845	-- 5
.11479	.01990	.05862	-- 6
.11464	.02012	.05883	-- 7
.11459	.02034	.05902	-- 8
.11438	.02032	.05904	-- 9
.11392	.02028	.05907	--10
.11321	.02021	.05911	--11
.11250	.02015	.05915	--12
.11204	.02010	.05918	--13
.11183	.02008	.05920	--14
.11181	.01979	.05894	--15
.11170	.01950	.05870	--16
.11148	.01924	.05851	--17
.11114	.01909	.05842	--18
.11078	.01915	.05853	--19
.11056	.01938	.05877	--20
.11047	.01966	.05902	--21
.11026	.01963	.05904	--22
.10980	.01959	.05907	--23
.10909	.01952	.05911	--24
.10839	.01945	.05915	--25
.10793	.01940	.05918	--26
.10772	.01938	.05920	--27
.10770	.01911	.05894	--28
.10758	.01884	.05869	--29
.10736	.01860	.05850	--30
.10702	.01847	.05843	--31
.10667	.01854	.05853	--32
.10645	.01874	.05876	--33
.10636	.01900	.05902	--34
.10615	.01898	.05904	--35
.10569	.01894	.05907	--36
.10498	.01887	.05911	--37
.10427	.01881	.05915	--38
.10381	.01877	.05918	--39
.10360	.01874	.05920	--40

TABLE 3.2 (continued)

```
   .10358       .01847       .05894       --41
   .10345       .01817       .05867       --42
   .10315       .01791       .05847       --43
   .10274       .01775       .05837       --44
   .10234       .01767       .05835       --45
   .10199       .01764       .05836       --46
   .10170       .01763       .05838       --47
   .10147       .01762       .05840       --48
   .10130       .01762       .05842       --49
 2      Type - EDGE NO:  3 ----------------
20.00     Tension parameter
 2      Number of nodes
   .11645       .01677       .05842       -- 1
   .11596       .01985       .05842       -- 2
 3      StretchType
 1.1000      Stretching parameter BETA
 2      Type - EDGE NO:  4 ----------------
20.00     Tension parameter
 2      Number of nodes
   .10181       .01439       .05842       -- 1
   .10130       .01762       .05842       -- 2
 3      StretchType
 1.1000      Stretching parameter BETA
 1      Type - EDGE NO:  5 ----------------
   .11737       .01347       .05530       -- 1
   .11719       .01345       .05530       -- 2
   .11688       .01342       .05531       -- 3
   .11642       .01337       .05532       -- 4
   .11590       .01332       .05533       -- 5
   .11545       .01327       .05533       -- 6
   .11513       .01324       .05534       -- 7
   .11495       .01322       .05534       -- 8
   .11486       .01353       .05565       -- 9
   .11458       .01375       .05591       --10
   .11416       .01376       .05598       --11
   .11381       .01354       .05581       --12
   .11363       .01324       .05554       --13
   .11359       .01292       .05523       --14
   .11338       .01290       .05525       --15
   .11303       .01287       .05526       --16
   .11251       .01283       .05529       --17
   .11192       .01278       .05531       --18
   .11140       .01274       .05534       --19
   .11105       .01271       .05536       --20
   .11084       .01269       .05536       --21
   .11074       .01297       .05567       --22
   .11046       .01317       .05592       --23
   .11004       .01318       .05598       --24
   .10970       .01298       .05581       --25
   .10952       .01270       .05554       --26
   .10948       .01241       .05523       --27
   .10927       .01239       .05525       --28
```

TABLE 3.2 (continued)

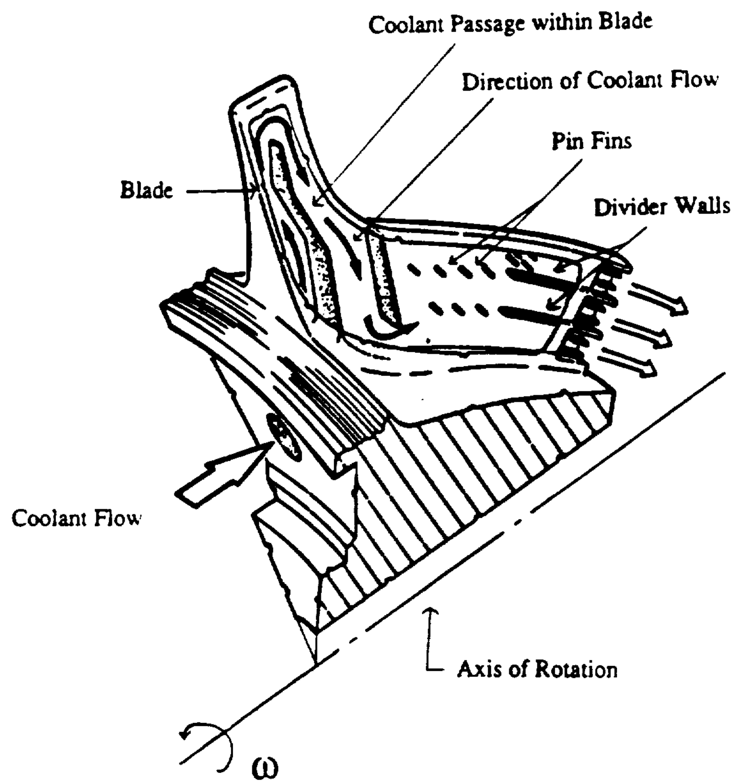
.10892	.01235	.05526	--29
.10840	.01231	.05529	--30
.10780	.01225	.05531	--31
.10728	.01221	.05534	--32
.10693	.01217	.05536	--33
.10672	.01215	.05536	--34
.10663	.01242	.05565	--35
.10639	.01262	.05589	--36
.10601	.01266	.05598	--37
.10567	.01250	.05588	--38
.10545	.01225	.05566	--39
.10536	.01197	.05539	--40
.10519	.01197	.05541	--41
.10494	.01197	.05545	--42
.10459	.01196	.05549	--43
.10418	.01195	.05555	--44
.10373	.01194	.05561	--45
.10332	.01193	.05567	--46
.10297	.01192	.05572	--47
.10272	.01191	.05576	--48
.10256	.01195	.05582	--49
1 Type -	EDGE NO:	6	----------------
.11697	.01656	.05530	-- 1
.11682	.01654	.05530	-- 2
.11656	.01652	.05531	-- 3
.11618	.01648	.05532	-- 4
.11574	.01643	.05533	-- 5
.11536	.01639	.05533	-- 6
.11510	.01637	.05534	-- 7
.11495	.01635	.05534	-- 8
.11486	.01666	.05565	-- 9
.11458	.01689	.05591	--10
.11416	.01690	.05598	--11
.11381	.01668	.05581	--12
.11363	.01637	.05554	--13
.11359	.01605	.05523	--14
.11338	.01603	.05525	--15
.11303	.01600	.05526	--16
.11251	.01596	.05529	--17
.11192	.01591	.05531	--18
.11140	.01587	.05534	--19
.11105	.01584	.05536	--20
.11084	.01583	.05536	--21
.11074	.01610	.05567	--22
.11046	.01631	.05592	--23
.11004	.01632	.05598	--24
.10970	.01613	.05581	--25
.10952	.01585	.05554	--26
.10948	.01555	.05523	--27
.10927	.01554	.05525	--28
.10892	.01551	.05526	--29
.10840	.01547	.05529	--30

TABLE 3.2 (continued)

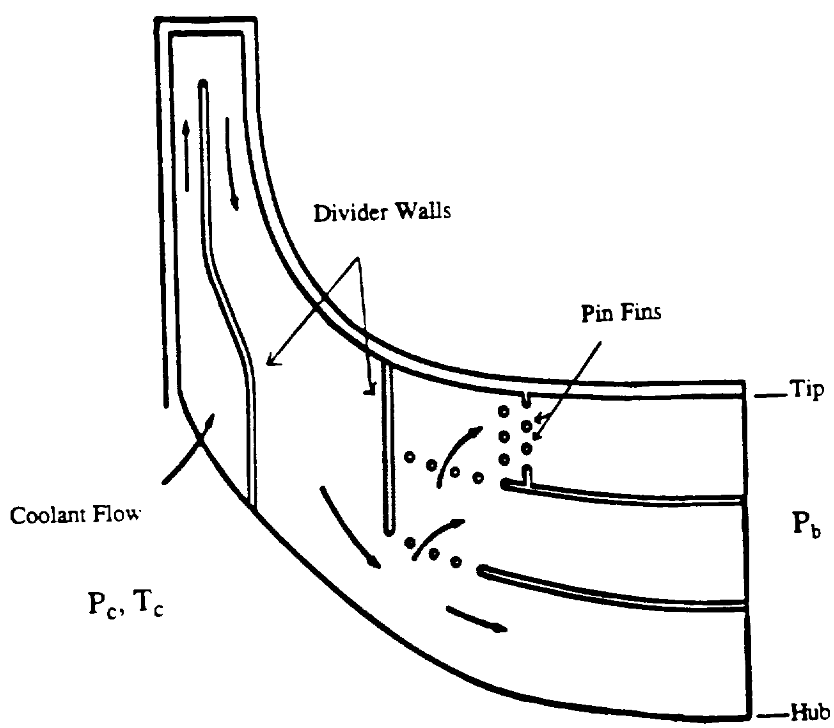
```
   .10780      .01543      .05531      --31
   .10728      .01539      .05534      --32
   .10693      .01536      .05536      --33
   .10672      .01534      .05537      --34
   .10663      .01561      .05565      --35
   .10639      .01581      .05589      --36
   .10602      .01586      .05598      --37
   .10567      .01571      .05588      --38
   .10545      .01546      .05565      --39
   .10536      .01519      .05538      --40
   .10516      .01519      .05541      --41
   .10487      .01519      .05544      --42
   .10448      .01518      .05549      --43
   .10400      .01518      .05555      --44
   .10348      .01518      .05561      --45
   .10300      .01518      .05567      --46
   .10261      .01518      .05572      --47
   .10232      .01518      .05576      --48
   .10212      .01522      .05582      --49
 2      Type - EDGE NO:  7 -----------------
20.00     Tension parameter
 2      Number of nodes
   .11737      .01347      .05530      -- 1
   .11697      .01656      .05530      -- 2
 3      StretchType
 1.1000      Stretching parameter BETA
 2      Type - EDGE NO:  8 -----------------
20.00     Tension parameter
 2      Number of nodes
   .10256      .01195      .05582      -- 1
   .10212      .01522      .05582      -- 2
 3      StretchType
 1.1000      Stretching parameter BETA
 2      Type - EDGE NO:  9 -----------------
20.00     Tension parameter
 2      Number of nodes
   .11645      .01677      .05842      -- 1
   .11596      .01985      .05842      -- 2
 3      StretchType
 1.1000      Stretching parameter BETA
 2      Type - EDGE NO: 10 -----------------
20.00     Tension parameter
 2      Number of nodes
   .11737      .01347      .05530      -- 1
   .11697      .01656      .05530      -- 2
 3      StretchType
 1.1000      Stretching parameter BETA
```

TABLE 3.2 (concluded)

```
  2      Type - EDGE NO: 11 -----------------
 20.00     Tension parameter
  4      Number of nodes
    .11645      .01677      .05842      -- 1
    .11658      .01628      .05799      -- 2
    .11719      .01407      .05591      -- 3
    .11737      .01347      .05530      -- 4
  3      StretchType
  1.1000      Stretching parameter BETA
  2      Type - EDGE NO: 12 -----------------
 20.00     Tension parameter
  4      Number of nodes
    .11596      .01985      .05842      -- 1
    .11610      .01936      .05799      -- 2
    .11678      .01716      .05591      -- 3
    .11697      .01656      .05530      -- 4
  3      StretchType
  1.1000      Stretching parameter BETA
  2      Type - EDGE NO: 13 -----------------
 20.00     Tension parameter
  2      Number of nodes
    .10181      .01439      .05842      -- 1
    .10130      .01762      .05842      -- 2
  3      StretchType
  1.1000      Stretching parameter BETA
  2      Type - EDGE NO: 14 -----------------
 20.00     Tension parameter
  2      Number of nodes
    .10256      .01195      .05582      -- 1
    .10212      .01522      .05582      -- 2
  3      StretchType
  1.1000      Stretching parameter BETA
  2      Type - EDGE NO: 15 -----------------
100.00     Tension parameter
  3      Number of nodes
    .10181      .01439      .05842      -- 1
    .10253      .01226      .05613      -- 2
    .10256      .01195      .05582      -- 3
  3      StretchType
  1.1000      Stretching parameter BETA
  2      Type - EDGE NO: 16 -----------------
100.00     Tension parameter
  3      Number of nodes
    .10130      .01762      .05842      -- 1
    .10209      .01552      .05613      -- 2
    .10212      .01522      .05582      -- 3
  3      StretchType
  1.1000      Stretching parameter BETA
```



(a) Cutout view



(b) Schematic drawing

Figure 1.1 — Radial turbine coolant passage.

n (Technique: n = 2 or n = 4)
IL
JL
KL
σ_ξ
σ_η
σ_ζ
$k_{\xi 1}$
$k_{\xi 2}$
$k_{\eta 1}$
$k_{\eta 2}$
$k_{\zeta 1}$
$k_{\zeta 2}$
Type-1

Information for Edge 1

Type-2

Information for Edge 2

Type-3

Information for Edge 3

.
.
.

Type-m

Information for Edge m

m=8 if n=2 (two boundary technique)
m=16 if n=4 (four boundary technique)

StretchType 11
Beta1 Beta 2
StretchType 12
Beta1 Beta 2
StretchType 15
Beta1 Beta 2
StretchType 16
Beta1 Beta 2

Figure 3.1 — Input-file format for GRID3D-v2.

Information for Edge i

if Type-i = 1:

$x_1\ y_1\ z_1$ $x_2\ y_2\ z_2$ $\vdots$ $x_{NL}\ y_{NL}\ z_{NL}$	NL=IL for i=3,4,9,10,13 and 14 NL=JL for i=11,12,15 and 16 NL=KL for i=1,2,3,4

if Type-i = 2:

σ NP $x_1\ y_1\ z_1$ $x_2\ y_2\ z_2$ $\vdots$ $x_{NP}\ y_{NP}\ z_{NP}$ StretchType i Beta1 Beta2	σ = tension for spline NP = number of node points

Figure 3.1 (concluded)

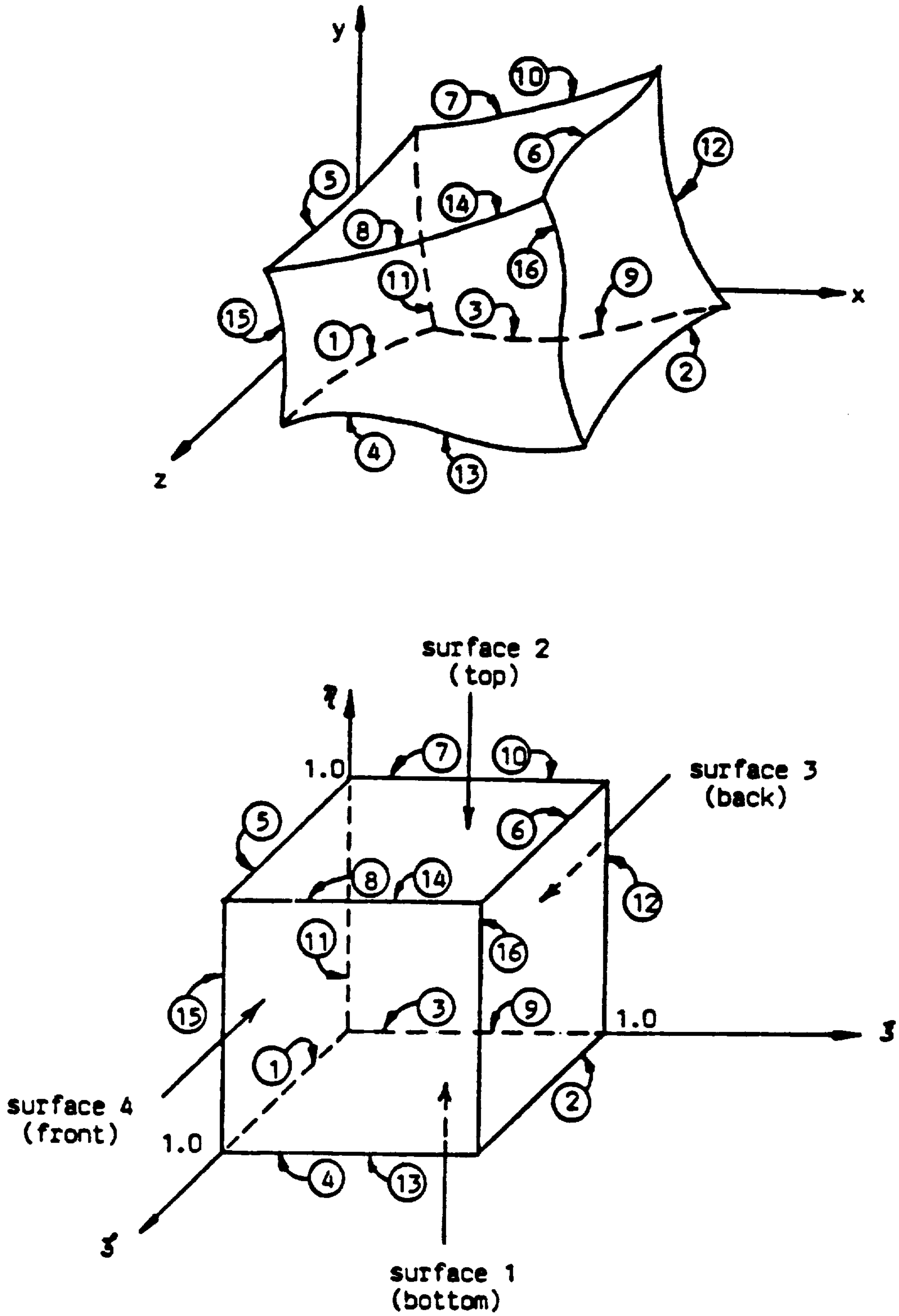


Figure 3.2 — Edge curve and boundary surface numbering scheme for GRID3D-v2.

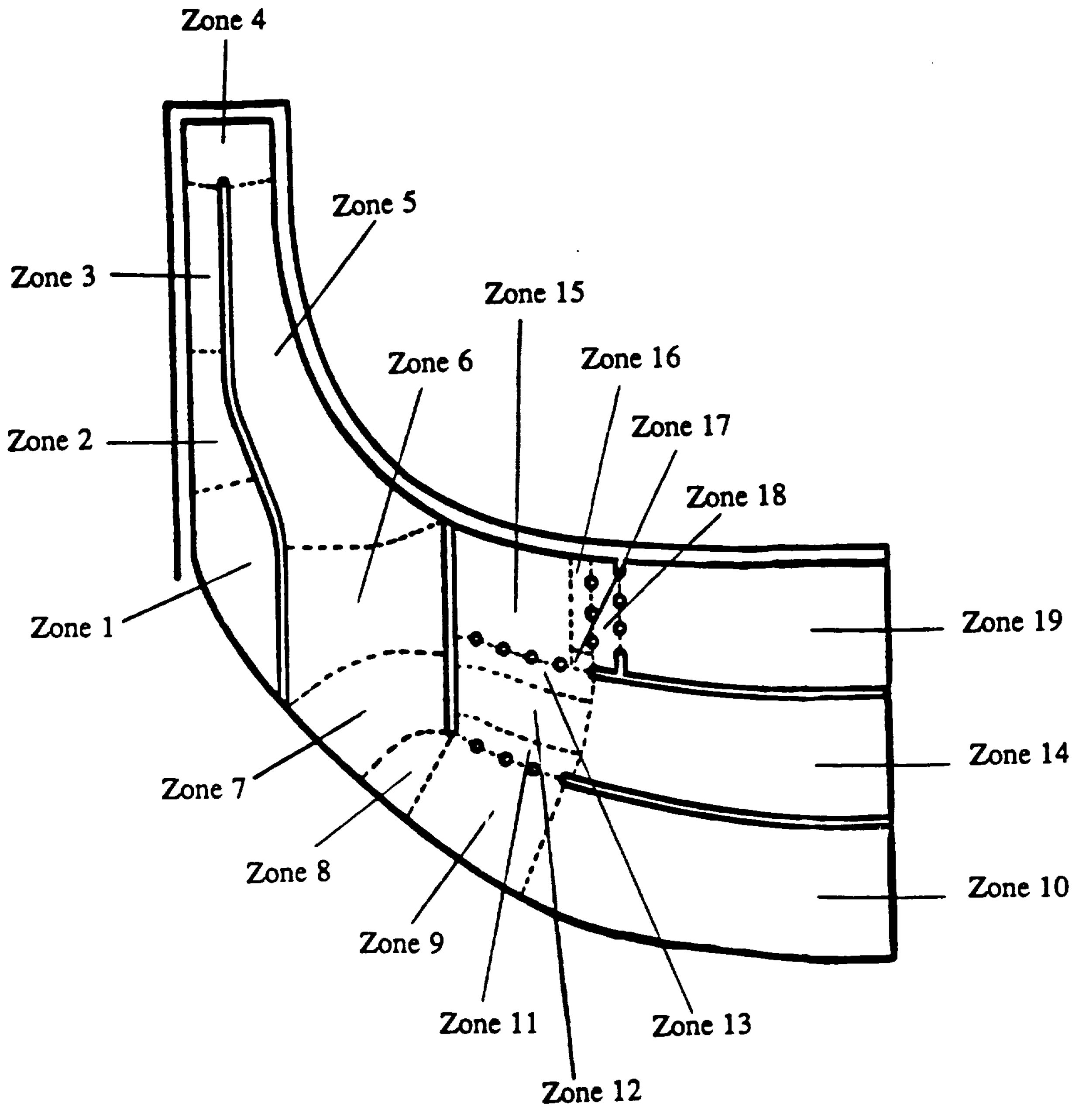


Figure 3.3 — Partitioning of the spatial domain of radial turbine coolant passage into zones for grid generation.

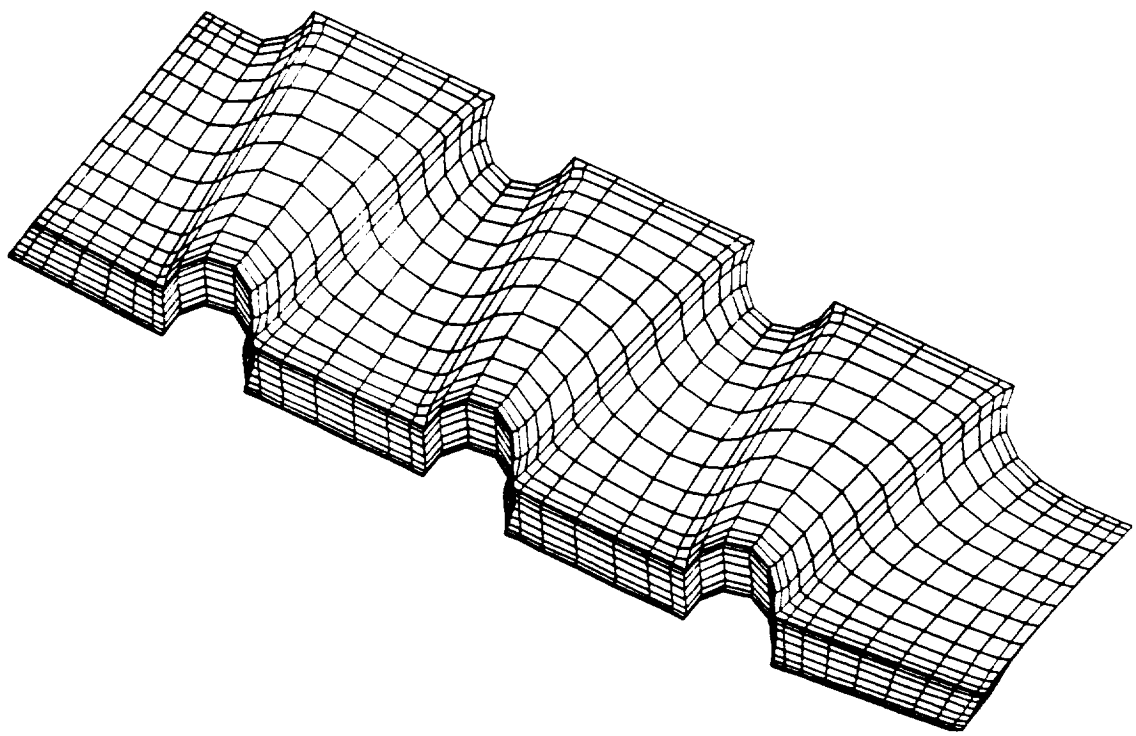


Figure 3.4 — Grid system for zone 18 of radial turbine coolant passage.

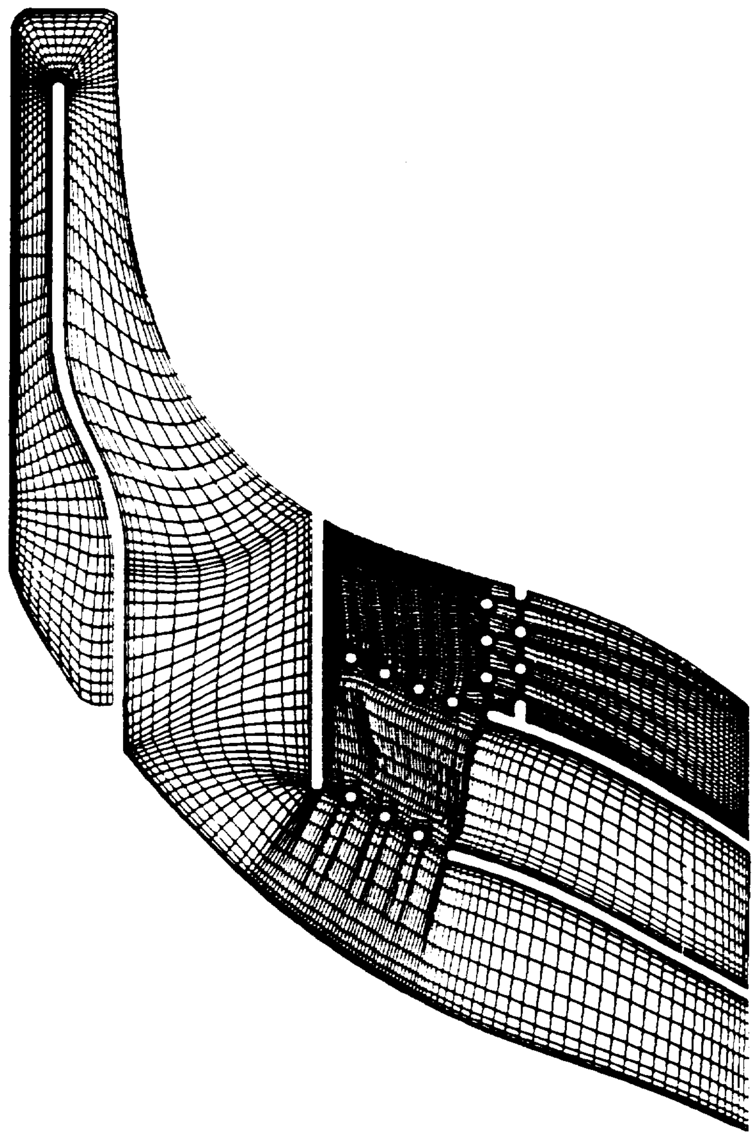


Figure 3.5 — Grid system for the whole radial turbine coolant passage (2-D view).

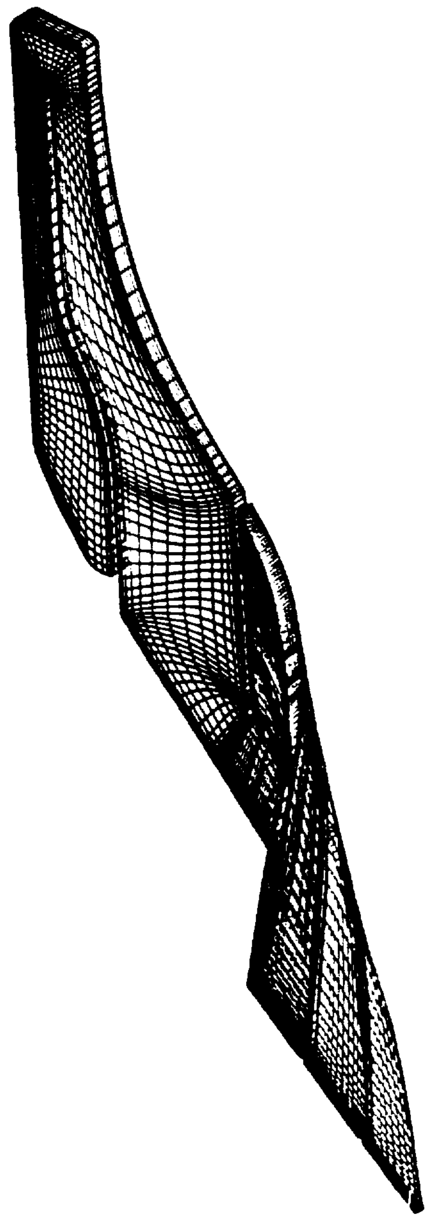


Figure 3.6 — Grid system for the whole radial turbine coolant passage (3-D view).

Report Documentation Page

NASA
National Aeronautics and Space Administration

1. Report No. NASA TM-103766	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle
GRID3D-v2: An Updated Version of the GRID2D/3D Computer Program for Generating Grid Systems in Complex-Shaped Three-Dimensional Spatial Domains

5. Report Date
June 1991

6. Performing Organization Code

7. Author(s)
E. Steinthorsson, T.I-P. Shih, and R.J. Roelke

8. Performing Organization Report No.
E-6028

10. Work Unit No.
535-05-10

9. Performing Organization Name and Address
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

11. Contract or Grant No.

13. Type of Report and Period Covered
Technical Memorandum

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Washington, D.C. 20546-0001

14. Sponsoring Agency Code

15. Supplementary Notes
E. Steinthorsson and T. I-P. Shih, Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213; R.J. Roelke, NASA Lewis Research Center. Responsible person, R.J. Roelke, (216) 433-3403.

16. Abstract
In order to generate good quality grid systems for complicated three-dimensional spatial domains, the grid-generation method used must be able to exert rather precise controls over grid-point distributions. In this report, several techniques are presented that enhance control of grid-point distribution for a class of algebraic grid-generation methods known as the two-, four-, and six-boundary methods. These techniques include variable stretching functions from bilinear interpolation, interpolating functions based on tension splines, and normalized "K-factors." The techniques developed in this study were incorporated into a new version of GRID3D called GRID3D-v2. The usefulness of GRID3D-v2 was demonstrated by using it to generate a three-dimensional grid system in the coolant passage of a radial turbine blade with serpentine channels and pin fins.

17. Key Words (Suggested by Author(s))
Grid generation

18. Distribution Statement
Unclassified - Unlimited
Subject Category 61

19. Security Classif. (of the report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified	112	A06

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161